MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DTIC
ELECTE
FEB 13 1986
S
D

A NETWORK MONITORING FACILITY

FOR A DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

THESIS

AFIT/GCS/EE/85D-14   JANICE F. ROWE
Captain   USAF

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DTIC
S ELECTE D
FEB 1 3 1986
D

A NETWORK MONITORING FACILITY

FOR A DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

THESIS

AFIT/GCS/EE/85D-14   JANICE F. ROWE
                          Captain   USAF

NETWORK MONITORING FACILITY

FOR A

DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Masters of Science

Janice F. Rowe, B.S.

Captain, USAF

December 1985

Accesion For		
NTIS CRA&I	☒	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and/or Special	
A-1		

Approved for public release; distribution unlimited

Preface

The purpose of this study was to design and implement a performance monitor for a Distributed Data Base Management System. Although many of the articles and papers read as preparation for this study mentioned the need for such a monitor, none contained a detailed examination of what was to be measured and how the measurements were obtained or presented a methodology for determining how to conduct such an examination. Capt Paul D. Bailor, presented with the same problem in his attempt to develop a Data Base Management System Performance Monitor, conducted a detailed analysis of the DDBMS performance evaluation process and included this analysis in his thesis report. This reference proved invaluable and I would like to thank Capt Bailor for the completeness of his work, it provided an excellent guideline for my study.

I would also like to thank my thesis advisor, Dr Thomas Hartrum for all his help and encouragement as well as the other members of my thesis committee, Dr Gary Lamont and Major Walter Seward, for their help in preparing this report. Finally, I wish to thank my husband Mark, who though faced with a thesis effort of his, own found the time to support and encourage me in mine.

Janice F. Rowe

ii

Table of Contents

VOLUME I

Note: Volume II and Volume III are maintained by Dr. Thomas A. Hartrum,
AFIT/EN.

List of Figures

List of Tables

Abstract

This investigation designed and implemented a hybrid network monitoring facility on an existing Distributed Data Base Management System. Analysis of the performance evaluations goals and objectives for the complete distributed system (both the layered protocol network and distributed data base) was accomplished. Two monitoring programs were developed. The on-line analysis monitor is designed to work with existing software to calculate metrics involving arrival rates, packet counts and arrival times. The off-line analysis monitor, using the packets saved during the on-line session, completes a more detailed analysis, providing user selectable metrics in the area of throughput, response times and utilization. Both programs were extensively tested using a four phased process which encompassed unit level, integration, systems and operational testing. Operational testing was accomplished using an artifical traffic generator program, designed to produce realistic network traffic.

I. Introduction

Over the past decade, the trend in computer systems has been away
from centralized computing systems and towards distributed data pro-
cessing systems. These systems are characterized by the location of
their resources and data at different work sites. Microcomputers,
serving as user workstations as well as storage devices and printers are
distributed to the work areas and connected to other similar work-
stations. System's software and data is shared among all users,
increasing the level of complexity for such systems. Since the system's
resources are shared among several users, it requires a centralized
management function to insure efficient and effective operation. This
management function relies heavily upon performance evaluation to accom-
plish its goals. However, the development of performance evaluation
techniques has not kept up with the development of new distributed
systems (2:388). Little work has been done in the area of performance
evaluation for distributed systems beyond the simulation and modeling of
computer networks. Actual performance monitoring is a necessary step
since it can indicate what the system is doing, a prerequisite to under-
stand "why" (19:2).

A Historical Perspective

Before examining distributed data networks and current methods used
in their performance evaluation, it is necessary to understand the back-
ground of these systems and the changes that have occurred in the
computer field over the past twenty years. Early computers were large,
expensive, complicated machines, characterized by a relatively slow

1

operating speed, vacuum tube construction, high electrical consumption
and special environmental requirements (temperature, humidity, etc).
Referred to as the first generation, these computers performed one opera-
tion at a time, lacking the capability for simultaneous operations
(10:525-530). The software on these machines was rudimentary, usually
consisting of a program loader and some simple utility programs. Periph-
eral devices consisted of a card reader,and a line printer. Jobs on the
system were run in batch mode usually by the programmer. Due to hardware
costs involved and the simplistic nature of the software, computer per-
formance evaluation centered around hardware and was based on such para-
meters as CPU cycle time and instruction execution time (21:2-3).

Computers gradually evolved over the next ten years. The intro-
duction of larger, less expensive memories, new hardware technology,
better peripheral devices, and improved software aids made computers
easier to use. This second generation can be categorized as having
increased reliability and reduced environment requirements due to
transistor construction (10:525-530). These computers allowed simul-
taneous operations, interactive access and more complicated software.
This software included the introduction of operating systems which pro-
vided a uniform environment for writing and running programs. Hardware
costs far exceeded those of software, so performance evaluation still
centered around hardware performance with some consideration given to
system throughput (21:2-4).

Characterized by integrated circuits, the third generation computers
were smaller, cheaper and faster than those which preceded it (10:532-
554). The computer organization was modular with units, (processor, main

memory and data channels), added or removed to meet changing require-
ments. The third generation saw the introduction of telecommunications,
though early computer networks were mainly used to allow users at one
site to log onto a computer at another site or allow users to transfer
files between sites (9:4-5). Software was changed drastically. Compli-
cated operating systems and control programs were needed to coordinate
the complex hardware configuration, efficiently use multiprogramming
capabilities, and handle all the terminal communication requirements
(10:1353-1358). The nature of performance evaluation also changed
dramatically. In an article by Henry Lucas, the changing nature was
explained as follows:

> "Because the programming system is an integral part of modern
> computers, the evaluation process must now consider software as well
> as hardware in assessing performance. The capabilities of the
> operating system are central to the performance of the computer:
> particularly crucial are any multiprogramming and multiprocessing
> features. The speed of assembly and compilation plus the execution
> of the resultant output code are also of great importance.
> Application programs, special telecommunications packages, and
> utilities are also part of the computer system and their performance
> is a component of total systems performance." (16:79)

The fourth generation has seen the introduction of large and very
large scale integrated circuit design and with it the introduction of
small, inexpensive, highly versatile, mini and micro computers (9:4-5).
These computers are characterized by an increase in abilities, speed and
efficiency, a reduction in costs and elimination of the special environ-
mental requirements of large systems. Improved telecommunications tech-
nology has allowed minicomputers to be placed in the work area with the
users (9:4-6). Microcomputers as well as system resources such as
memory, disks and printers are now distributed to the user work areas and
interconnected together, replacing the central computer. While hardware

costs have decreased with the introduction of new technology, software costs have soared (2:79). A system's software can now be expected to incur as much as eighty percent of the total systems initial development costs (24). The reason for this rise is the nature of the software required for distributed systems. The software is much more complex, requiring coordination of the tasks occurring at all nodes of the network (10:1359-1362). Additionally, distributed data base management systems are needed so the capabilities of the entire network can be available to each user. The complexity of distributed computer systems causes much of the software to contain gross inefficiencies that are not suspected by the designer (19:1-2). It is the goal of performance evaluation in fourth generation computer systems to detect these inefficiencies. Performance evaluation techniques also need to be changed to reflect the changing nature of computer systems and the increasing effect of communications.

Background

This thesis effort was impacted by two different areas, Data Base Management and Local Area Networks. Basic concepts in these areas are described.

Data Base Management. A data base is a collection of interrelated data, designed to be used by one or more applications, stored so that the data is independent of the applications which use it, and organized for rapid retrieval and processing (15:11). The software that manages and manipulates the data contained within the data base is known as the Data Base Management System (DBMS). A data base model is used by the DBMS

4

to describe the logical structure of the data base and how it is
processed (15:21). Three of the most commonly used models are:

A. Relational: Based on the theory of relational mathematics, the
data is represented by two dimensional tables (15:149-150). This
model is illustrated in Figure 1a.

B. Hierarchical: The data is represented as a hierarchy of
elements, organized in a tree-like structure composed of nodes and
links (15:61-62). In this structure, shown in Figure 1b, a node may
have exactly one parent but any number of children.

C. Network: The data is represented by a collection of nodes and
links as in the hierarchical model, but a node may have multiple
parents (15:69-70). Figure 1c illustrates this model.

The data base and its management system, when implemented on a
single computer, are referred to as centralized systems while those
spread over a network of computers are called distributed systems. Each
type of system has its own advantages and disadvantages. In a centra-
lized system, some costs (notably personnel and communications) are
reduced, better control is exercised over the data (especially in the
areas of security and data integrity), and the data is more available to
top-level management (8:5-6). The major constraints of a centralized
system are that the system may not be responsive to the user and that it
is more difficult to take advantage of rapidly changing technology. In a
distributed system these disadvantages are avoided. A distributed system
is more responsive since the computers are usually located in the user's
work area, more reliable since a single failure generally will not
prevent the network from functioning in a degraded mode, more economical

5

COURSE	DEPT	ROOM	INSTRUCTOR	UNITS
EE7.54	ENG	262	SEWARD	4
EE6.90	ENG	164	HARTRUM	2
MA5.31	MATH	162	LAULISS	4
EE4.50	ENG	60	WOFFINGTON	4
MA4.50	MATH	262	BROWN	3

A. RELATIONAL
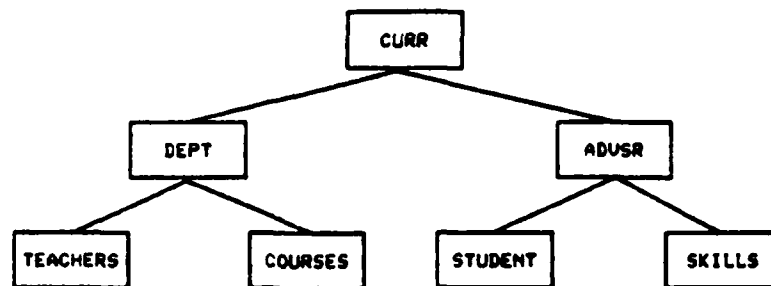
B. HIERARCHIAL
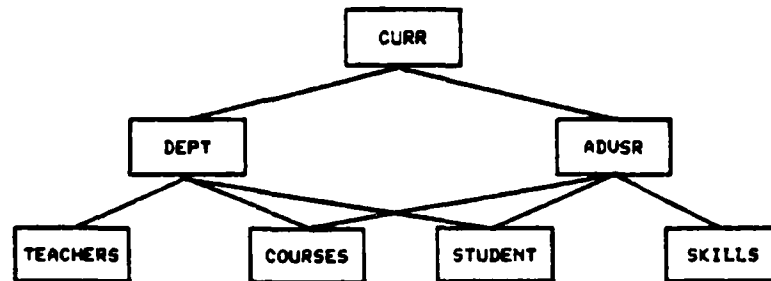
C. NETWORK

FIGURE 1. DATA BASE MODELS

since the cost of several small systems is usually less than that of a comparable large system; and more easily upgraded, either with replacement components or additional links and nodes (8:6-7). The major disadvantage of a distributed system is its increased complexity. A Distributed Data Base Management System (DDBMS) is more complicated and difficult to implement. More complex technology is needed to handle the increased problems of security, data integrity, recoverability and availability.

An example of a typical distributed data network (23) is shown in Figure 2. The system is made up of several components: the host sites equipped with local data bases and data dictionaries; network interfaces complete with extended data dictionaries and distributed data base management systems; and the actual network, the organization of equipment and lines that provide the communication service. A typical query (request for data) is serviced first at the local site. The local data directory is accessed for the location of the data. If the data cannot be found, the request is passed on to the network interface module where the query is translated into the DDBMS language and the extended data directory is accessed for the location of the data. If the location is found, the query is routed to that site and the response is passed back to the sending site. If the location cannot be found, a request for data location if sent to the central data directory which contains the location of all of the data items in the system. The central data directory finds the location of the data and sends that location to the requesting site which then routes the query to that site (23).

7
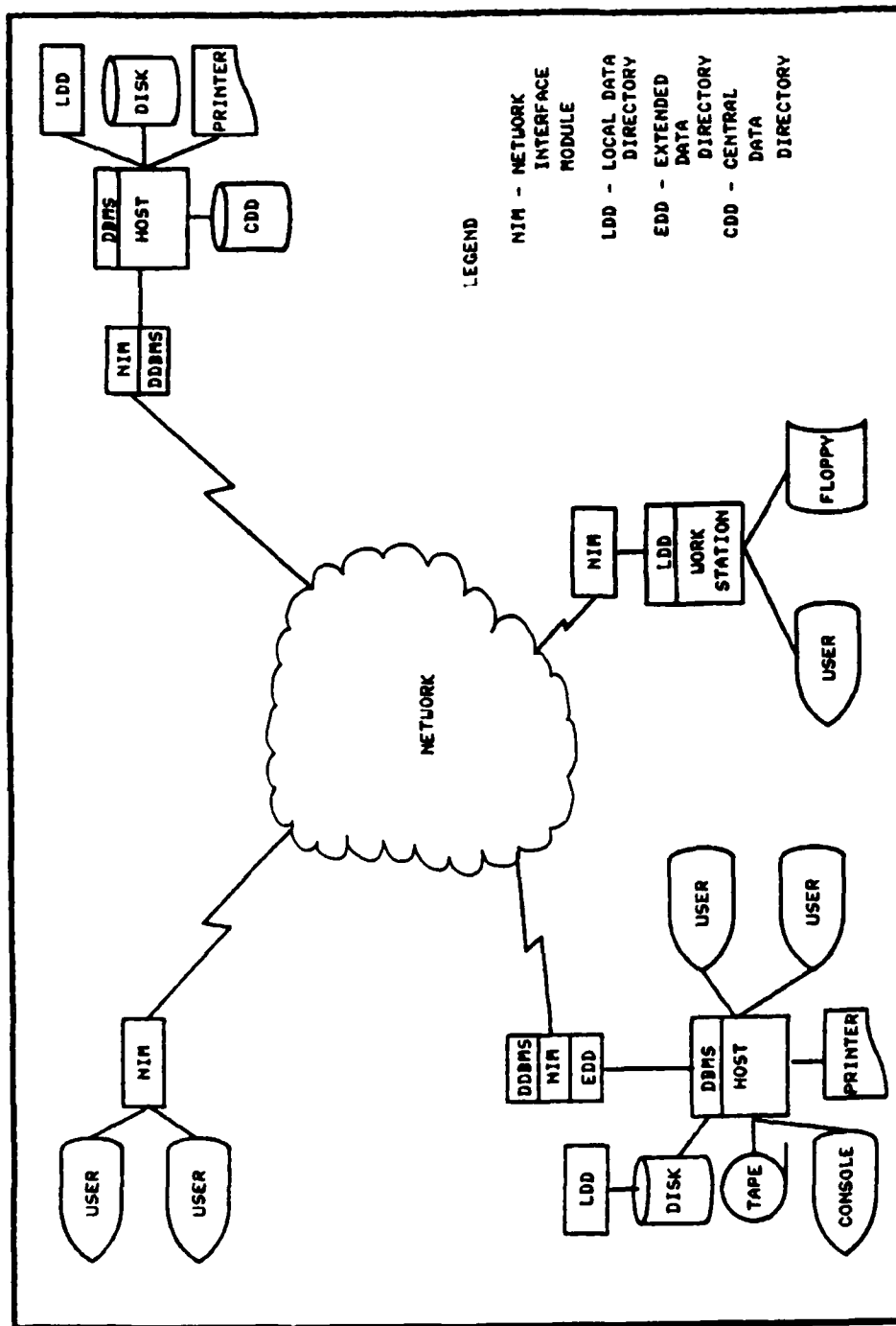
FIGURE 2. TYPICAL DISTRIBUTED DATA NETWORK

8

This is not to imply that every distributed system works in this manner. Some systems have eliminated local or extended data bases and all inquiries are routed through the centralized data directory. Other systems have complete data dictionaries at each site in the network. The important fact about distributed data networks is that all system inter-action is transparent to the user. For all intents and purposes, the user is logically, although perhaps not physically, accessing data on his local system (12).

Local Area Networks. A Local Area Network (LAN) is a communications system that provides service within a limited geographic area, usually with a radius of under a mile. These networks can span a building, campus or installation and are generally owned by a single entity (17:3). A LAN is designed to fit the nature of the distributed system it services and therefore the network can be constructed using a variety of topologies, the arrangement of links and nodes that make up the network. The most popular of these topologies are:

A. Star. All the sites in a Star network are joined at a single point (central site) and all routing is performed at this point (Figure 3a). This type of network is optimal when the traffic flow is predominantly from the central site to outer sites. Its biggest disadvantages are that the central site can be a single point of failure and the size and capacity of the network is a direct function of the size and capacity of the central site (9:34-35).

B. Ring. All the sites in this type of network are arranged to form a single unbroken ring (Figure 3b) and messages travel from site to site around the ring with each site retransmitting the

9
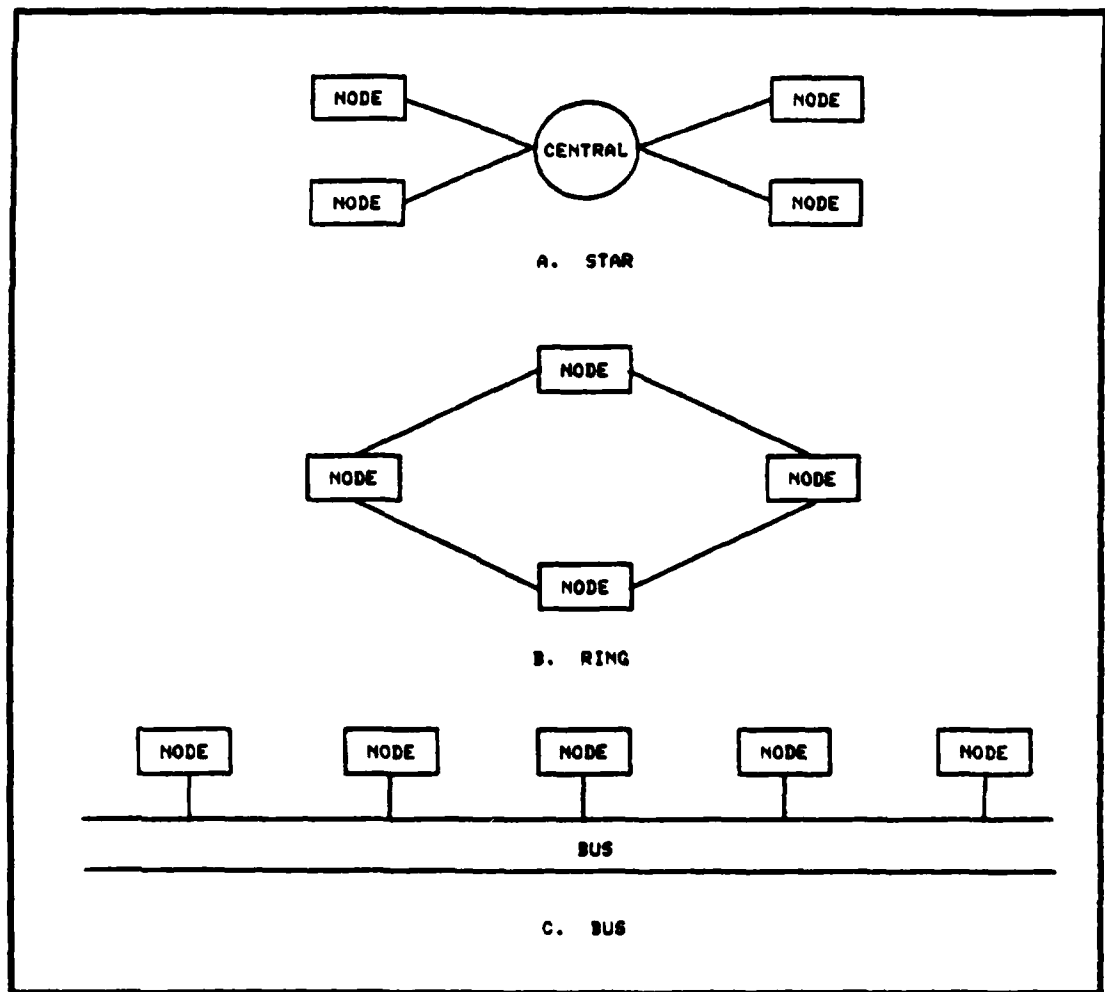
FIGURE 3. NETWORK TOPOLOGY

message to the next site. This increases the integrity of the information in the network since it is verified at each site. The disadvantage to a ring network is without bypass logic, any site can be a single point of failure. Installing this logic, increases the complexity of the system even further (9:35-37).

C. Bus. All the sites are multidropped and share a single, fully connected channel (Figure 3c). Messages are broadcast to all sites with each site responsible for recognizing its address and receiving its messages. This is the most popular type of network since a single site's failure doesn't effect the entire ring, there is no store-and-forward delay and the network is easily configured and expanded. Its primary disadvantages are the collisions between network messages and a more difficult fault detection and isolation (9:38-40).

Summary of Current Knowledge

For a number of years now, the Air Force has been investigating the feasibility of using distributed data base networks to solve Air Force Problems. The United States Air Forces in Europe has contracted with Boeing Corporation to develop a distributed data network for their intelligence functions. The Strategic Air Command is developing their own network, SACDIN, designed to provide a link to their remote operating locations.

Due to the impact of these and other efforts, the Air Staff tasked Rome Air Development Center (RADC) to develop and coordinate the research and development activities in this area. RADC contracted for research into distributed data base networks with the Computer Corporation of

America. This contract resulted in the publication of a three volume paper which detailed the problems of distributed data networks and investigated possible solutions to these problems. Additionally, RADC has sponsored several thesis efforts at AFIT which investigated different aspects of distributed data base networks. Four of those efforts had a direct effect on this investigation.

The first was an effort conducted by Captain Eric F. Imker (7:1-5) in 1982 which produced a high level design of a DDBMS for use on the computers in the AFIT Digital Engineering Laboratory (DEL). The second, by Captain John G. Boeckman (5:1-8), modified and expanded Imker's efforts. He felt that the primary purpose of a distributed data base was to allow the user to access several data bases spread over a network as if they were one. His thesis effort produced the basic design and partial implementation of such a network. The DDBMS designed by Captain Boeckman allowed two relational DBMSs, Dbase II and Ingress, to communicate via the LSINET located in the AFIT DEL (Figure 4).

The third effort, conducted by Captain Paul D. Bailor (3:I-2 - I-5), developed the methodology for conducting a performance monitoring effort on a centralized DBMS and developing a generalized design for a corresponding DBMS software performance monitor. This effort included an extensive requirements analysis in the area of performance evaluation on centralized computer systems. The fourth, a continuation of Captain Bailor's effort, was conducted by Captain Timothy D. Bruner (6:I-1 - I-7). This effort continued Bailor's work by developing and implementing a user friendly interface to the existing DBMS performance monitor. This interface was intended to facilitate the job of the Data Base

12
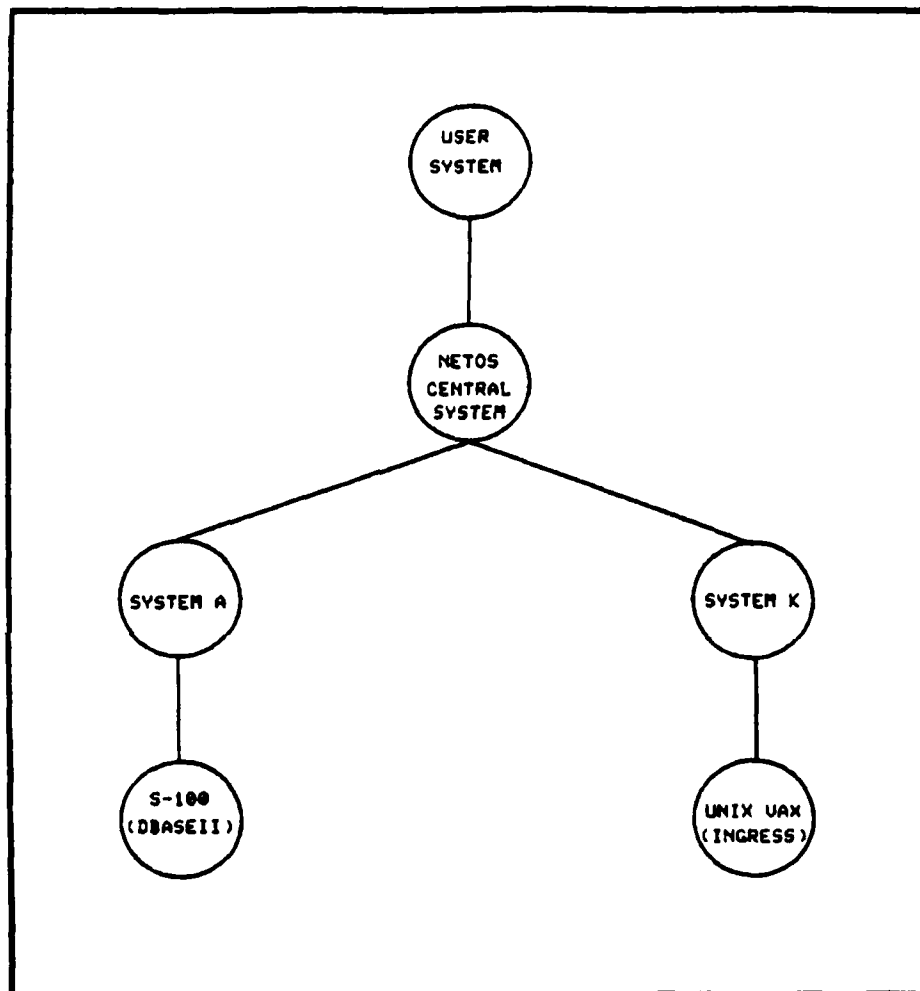
FIGURE 4. LSINET DDBMS CURRENT IMPLEMENTATION
(S1118)

Administrator by allowing the option of selecting the performance

parameters to be modified.

A fifth thesis effort that affected this study was conducted simul-

taneously with this effort by Capt James Wedertz. His effort was

directed at furthering the implementation of the DDBMS designed by

Captain Boeckman. He concentrated on fully implementing a centralized

data directory and to the extent possible his study was used to provide

LSINET DDBMS message formats and performance goals for this thesis

investigation.

Problem

The DDBMS software, as implemented on the LSINET, had no performance

metrics included. Therefore there was no means, other than visual

inspection, of evaluating the performance of the software or network.

Additionally, several other thesis efforts were underway that would have

benefited from DDBMS and network performance evaluation. Finally, since

the data base is a resource shared among several users, it required a

centralized management function to insure the DDBMS was efficiently and

effectively utilized. Questions that needed to be answered by this

management function included: where should the data be stored, how

should the data be organized, what functions of the data base need to be

distributed, how to insure data base integrity, and how responsive is the

network to the user. This management function would have been assisted

by the implementation of a network monitor. Therefore the goal of this

effort was to expand on Captains Imker's and Boeckman's efforts by

designing and implementing a network traffic monitor capable of:

A. generating real-time, on-line performance statistics.

14

B. generating off-line reports, summarizing performance statistics over a yet undetermined period of time.

C. handling the maximum traffic load of the network.

Approach

This thesis effort was accomplished in the following four phases:

A. Research

B. Requirement Analysis

C. Systems Design

D. Implementation and Testing

During the research phase, the various aspects of the problem were fully investigated. A review of previous thesis efforts as well as the published articles and books was conducted to become familiar with the concepts involved. The operations manuals for the LSI 11 micro-computer systems and the LSINET (14) as well as the available notes on the RT-11 Operating System and the Network Operating System (14) were also reviewed to become more familiar with the operation of the actual network.

In the requirements analysis phase, the actual performance metrics to be used were derived from the material reviewed in phase one and from other AFIT thesis students pursuing topics in related areas. The message traffic on the LSINET was analyzed to see how it could be used or modified to provide the required measurements. Also during this phase, decisions on the presentation of the metrics (on line, off-line, graphic, tabular), were made.

During the Systems Design phase the activities encompassing a performance evaluation monitor were determined and the processes required to accomplish these activities designed. Those processes included the On-

15

line Analysis process which represents the software module that will receive all traffic from the network, store it for off-line analysis and generate the on-line performance statistics and the Off-line Analysis Process which represents the software module that will generate the off-line performance statistics.

Finally in the implementation and test phase, the software processes designed during the system design phase were coded, integrated and tested. An artificial traffic generator was used to insure accurate and repeatable results and the software was modified as necessary.

Scope

This study developed a set of performance objectives for the LSINET DDBMS and generated the design of a performance monitor based upon those objectives. Every attempt was made to design a monitor general enough to be used with any type of standard network topology although this study only implemented the design on the current network (star topology). Every attempt was also made to fully implement an operational monitor for the LSINET DDBMS subject to the constraints given below.

Assumptions and Constraints

A. The LSINET DDBMS message format and the NETOS communications protocol frame format as shown in Appendix A, System Environment, correspond to the actual format of the packets transferred by the LSINET. Changes in either of these formats will necessitate a change in the monitor programs since they are both format dependent.

16

B. The LSINET DDBMS will be operational and capable of generating usable network traffic for the monitor. If this is not the case, operational acceptance testing of the monitor will be eliminated.

Materials and Equipment

This thesis effort required access to a dedicated workstation on the LSINET. This equipment was available in the AFIT DEL.

Overview of Thesis

The format of this report follows the approach detailed earlier. Chapter II provides a description of the requirements analysis conducted on the necessary functions of the network monitor. Chapter III presents the systems design of the Network monitor for use on the LSINET. Chapter IV describes the implementation procedures used and the testing conducted in the evaluation of the monitor. Chapter V summarizes the results of this thesis and recommends actions to be taken for further thesis efforts.

II. Requirements Analysis

Introduction

This chapter presents the results of the analysis performed on the problem of designing and implementing a Network Performance Monitor for the Digital Equipment Laboratories Distributed Data Base Management System (LSINET DDBMS). In order to satisfy the requirements of all readers, it is divided into two sections. The first deals with general concepts in the area of performance evaluation and presents a methodology for the process of developing a network performance monitor. Readers familiar with the general concepts involved in performance monitoring and evaluation may wish to skip to the second section which applies the concepts discussed in the first section to the problem of developing a performance monitor for the LSINET DDBMS.

General Concepts in Performance Evaluation

The first problem encountered in performance evaluation is the meaning of the word performance. Since performance is a qualitative measure and highly subjective to the needs of the people involved with the system, it has many different interpretations (21:8). Performance has been loosely defined as the effectiveness with which the resources of a system are utilized toward meeting its objectives (21:8). That definition, loosely paraphrased as "how well the system does what it was intended to do", will suffice for this thesis.

Based on this definition, performance evaluation can be defined as the process of collecting and analyzing a system's performance information and comparing it to that system's desired capabilities. In the case of a distributed data network, this task becomes more difficult since the

system referred to is the entire distributed network including the hardware and software at each site as well as the organization of lines and equipment that make up the communications network. Each of these components must be examined individually to determine its individual performance and the system must then be evaluated as a whole to determine overall performance. There are basically four steps in the process of performance evaluation: determine the goal of the evaluation, determine what to measure, determine how to measure it, and analyze the results of the measurement (11:26-32). Each step is described in more detail below.

Determine the Goal of the Performance Evaluation. The main purpose of the measurement and evaluation of a computer or communication system are: to aid in the design of hardware and software, to aid in the system selection process, and to provide data on the actual performance of an existing system (16:79-81). The first two purposes are accomplished primarily through simulation and modeling and are not included in the scope of this thesis. The third purpose is usually accomplished through performance monitoring, a method of collecting data on the performance of an existing system through the prolonged observation of the system's behavior (21:79). Once the purpose of the evaluation has been established, the actual goals of that evaluation can be determined. The general goal of performance monitoring is to gain an insight into system behavior either to improve performance or to add functionality. This general goal can be divided into more specific goals such as: evaluate network traffic and characteristics, lower costs, determine level of performance, support network management and determine availability and reliability.

Determine What to Measure. Once the goals of the evaluation are established, the next step is to determine the actual performance measures (21:76). These measures are influenced by two criteria, the purpose of the system and the viewpoint of the evaluation (19:3-5).The purpose of the system refers to what the system's role is. Performance of a system can be discussed only in the context of what the system is required to do (21:10). For example, if the system is a real time, fee-for-service network, response time may become the primary performance measure. If however, it is installed in a hospital, reliability might be more important. The second criteria is the viewpoint of the evaluation. Measurement evaluated from the user's viewpoint might be more concerned with the function of an individual site as it relates to the entire system, i.e. response time. Measurement evaluated at the system administrator level would be more concerned with the function of the overall system, and measures such as throughput and utilization might be considered more important.

Generally, there are three classic performance measures used to evaluate computer systems: throughput (task completions per time unit); response time (interval of time starting when a user submitted a request and ending when a response is obtained); and resource utilization (percentage of time the resource is in use). As seen in Table I, these measures are applicable at each level of a distributed data network. These measures encompass a great many individual performance metrics, discussed in detail in the second section of this chapter.

Determine How to Measure It. The third step in performance evaluation is to determine how to obtain data on the selected performance

TABLE I

Classic Performance Metrics (16:79)

Level: Parameter	Description
NETWORK:	
Throughput	Number of messages transmitted per unit time
Response Time	Interval of time starting when a message enters the network and ending when an acknowledgement is received by the sending node
Utilization	Percentage of time each of the nodes and links are in use
DBMS:	
Throughput	Number of queries executed at a local DBMS per unit time
Response Time	Interval of time starting when a query is submitted to the local DBMS and ending when a reply is returned
Utilization	Percentage of time each of the local DBMSs are is use
DDBMS:	
Throughput	Number of queries executed by the distributed system per unit time
Response Time	Interval of time starting when the user submits a query and ending when a response is received by the user
Utilization	Percentage of time each of the components in the Distributed Network is in use

metrics. As mentioned above, performance monitoring is the method of collecting data on the actual performance of an existing system (21:79-80). A performance monitor is the tool that facilitates the performance analysis and evaluation. Its main functions are event detection, data collection, data reduction, and presentation of results (10:362). A performance monitor usually falls into one of three categories: software, hardware or hybrid.

A software monitor is a program, incorporated into the operating system or applications program, that is capable of measuring the performance of either computer systems or computer programs (10:1362). This type of monitor can either be of the event driven or sampling type. An event driven monitor is activated by a specific event, an instruction or flag for instance. Upon the occurrence of this condition, the monitor is activated, the event noted and the monitor deactivated until the next occurrence (10:1362). A sampling monitor is similar to an event driven monitor but it is activated by an interval timer, collects data about the system for a set period of time and then is deactivated by a timer. A typical use for a software monitor can be seen in a job accounting system where the hardware resource utilization, memory utilization, amount of input/output, and CPU time are calculated for each job. The largest disadvantage of a software monitor lies in the fact that it introduces additional software overhead to the system and extra timing considerations must be made to insure that the monitor does not interfere with the normal operation of the computer system.

A hardware monitor is a device which measures electrical events (e.g. pulses, voltage levels) in a digital computer (10:679). It is used

22

to gather data for measurement and evaluation of computer systems, particularly computer hardware fault detection and isolation. A hardware monitor consists of several components including probes, logic circuits, counters, comparators and data transfer registers. As an example of use, the hardware monitor might be connected to measure the busy time for a CPU and I/O channel controller, and determine their overlap with an end goal of determining the efficiency to the multiprocessing system (10:680). The biggest disadvantage is that a hardware monitor only measures specific electrical events at predetermined points.

The disadvantages of the hardware and software monitors are solved in part by the hybrid monitor which combines elements of both the hardware and software monitors. Residing in a separate processor, the hybrid monitor can be fed data from the operating system of the computer as well as through probes. The major advantage of the hybrid monitor is that it allows accurate and comprehensive monitoring of the entire computer system. Its primary disadvantage is that it adds another level of complexity to an already complicated system (21:79).

Once the type of monitor is selected, a decision must also be made as to the type, centralized, distributed, or hybrid, of monitoring center. A centralized monitor appears as another site on the communications network (Figure 5-A). It must be capable of receiving all network message traffic (universal receiver). Additionally it must be capable of recognizing and responding to traffic routed specifically to it (i.e. update messages). The distributed center has connections to each site in the network (Figure 5-B). Completed messages are sent to this center along with the performance statistics for each message. The disadvantage

23
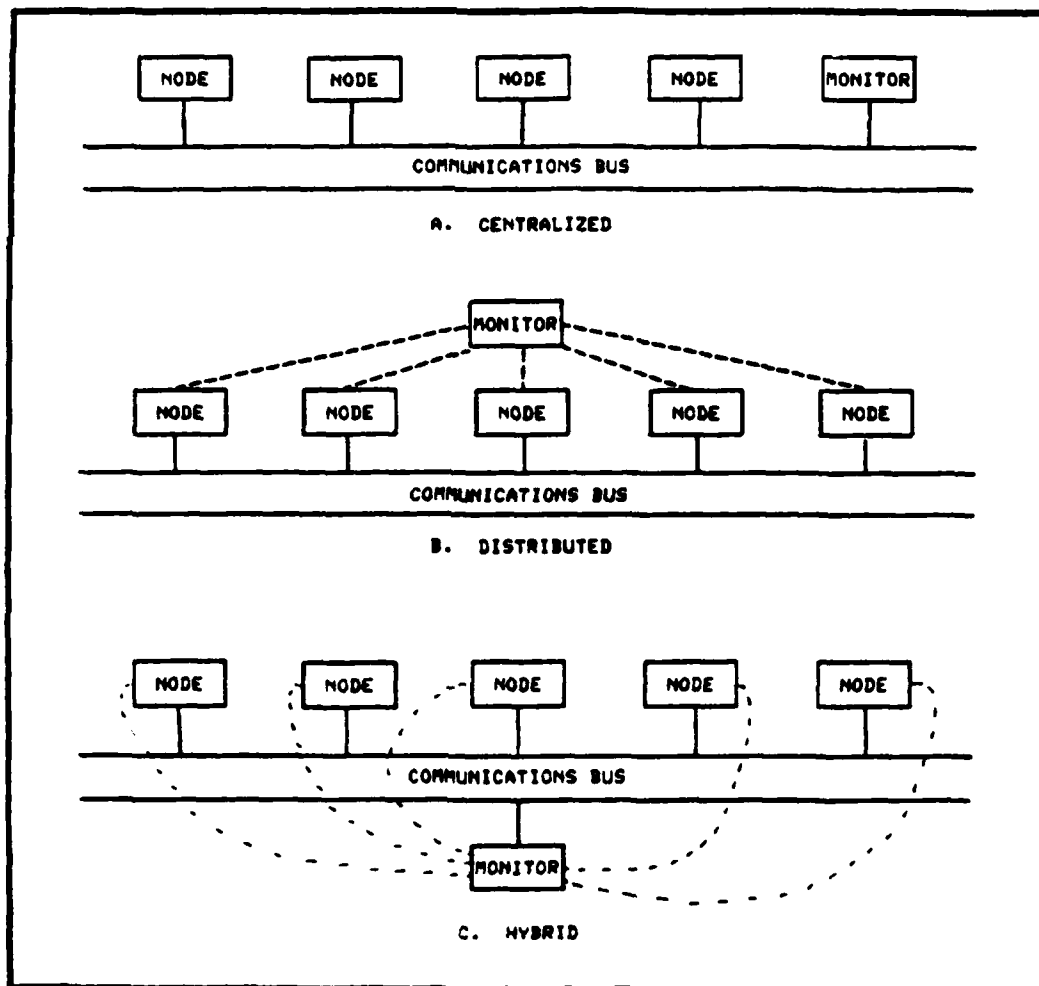
FIGURE 5. TYPES OF MONITORING CENTERS

of this type of center are the excessive communications involved to con-
nect each site in the network twice and the software overhead at each
site required to handle message flow (2:390). As in the case of types of
monitors, the problems of monitor location are largely solved by a hybrid
monitoring center (Figure 5-C). This type of center appears as another
site on the network and like the centralized monitoring center it must be
a universal receiver. Additionally, selected sites have monitoring soft-
ware installed at their nodes and with it perform local analysis on their
message traffic. The results of this local analysis are sent to the
hybrid monitor either via the network or over direct links where they are
combined with the rest of the analysis results. The advantage of the
hybrid monitor is that it presents a more complete analysis of the net-
work than is possible with either the centralized or distributed moni-
toring center. Its chief disadvantage is the increased software complex-
ity, the increased communications requirements and the increased delays.

Finally, the type of presentation, either on-line or off-line or
some combination of the two, must be considered. Several factors weigh
on this decision. First, if on-line presentation is used, decisions must
be made as to which performance metrics to present, how to present them
(graphically or via a report), how often to update the presentation (with
every new message or at specific time periods), and how to handle
incoming messages while the on-line system is being run. If an off-line
presentation is used, the decisions to be made include which performance
metrics to present, how to present them, how often to run the performance
monitor program.

25

Analyze the Results of the Measurement. Perhaps the most important step in the performance evaluation process is the analysis of the results of the measures. It is during this step that the conclusions are drawn from the graphs, tables and reports produced by the monitor. These conclusions consist of an evaluation of existing system performance and recommendations for system improvements. In short, they provide necessary feedback on actual system performance. The administrator compares these conclusions with the goals laid out in step one and determines what changes, either system tuning or upgrading, is needed to better achieve these goals.

Systems Requirements Analysis

This section details the results of the requirements analysis performed on the problem of developing a network performance monitor for the LSINET DDBMS. Readers unfamiliar with the specific environment, either the LSINET or the LSINET DDBMS and NETOS software that operate on the LSINET, are referred to Appendix A, System Environment. As described above, there are basically four steps in the process of performance evaluation: Determine the goal of the evaluation, determine what to measure, determine how to measure it and analyze the results of the measurements. In this part of the chapter, those steps have been applied to the problem and are discussed below. Prior to that discussion, however, a brief overview of previous studies and an analysis of the DDBMS performance evaluation process is presented.

Overview of Previous Studies. A review of the current literature available on the subjects of Distributed Networks, Distributed Data Bases, and Performance Evaluation for both Computer and Communication

Systems aided greatly in supplementing background information for this study, but produced little information on the problem of performance evaluation of a DDBMS. Captain Bailor, noting this problem in his thesis effort (3:II-24) conducted a detailed analysis of the DBMS performance process. That analysis is expanded in this study to cover the DDBMS performance process. Readers requiring more detailed information of the analysis process used are referred to Captain Bailor's study.

DDBMS Performance Analysis. The detailed analysis of the performance evaluation problem begins with an examination of the performance evaluation process for a DDBMS. This process is illustrated in Figure 6. In this figure, the users of the DDBMS generate the system inputs (DDBMS workload) and receive its output (completed work). The "service workload" process refers to general computer tasks such as responding to users' queries, running applications programs and printing reports and is not restricted to only DDBMS workload. The "determine DDBMS performance objectives" process is crucial to this study and is discussed in detail below. This process is a subset of the "DBMS performance evaluation" process which in turn is a subset of computer performance evaluation (3:II-30). The overall system is monitored to determine its effectiveness, capability to process a given workload, and efficiency, capability to process while minimizing resources used. Additionally, specific performance objectives for the DDBMS are established from the workload of the users, as well as the management requirements and objectives of the Network Manager or Data Base Administrator. These performance objectives along with the effectiveness and efficiency measures are inputs to the actual analysis process which uses these inputs to produce results for

27
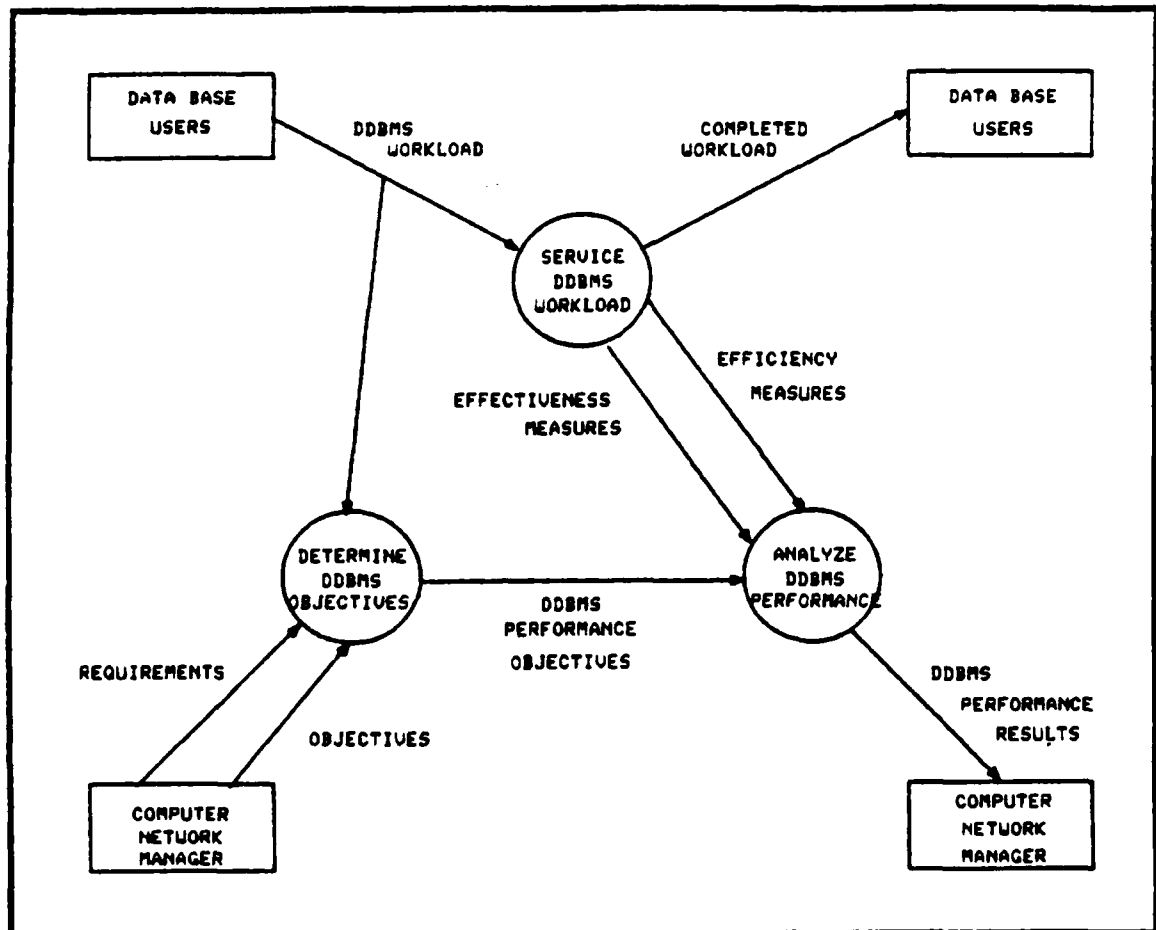
FIGURE 6. DDBMS PERFORMANCE EVALUATION PROCESS

(3. II-31)

28

the network manager. These results are used to determine how well the DDBMS is performing or to recommend changes to improve performance.

LSINET DDBMS Performance Objectives. As mentioned above, one of the most critical areas in performance evaluation is determining the system performance objectives. Without specific objectives, it is useless to monitor the performance of the DDBMS since there is no predetermined acceptable level or standard of comparison. The approach of "measure everything" is flawed since vast quantities of useless information are usually provided and must be sifted through to find the desired information and in most cases requires a prohibitive amount of storage (2:391). Also this method wastes system resources, computing and storing metrics that will never be used. A more goal oriented approach is preferred with the objectives of the analysis defined first and then the data and techniques required to meet the objectives determined (13:1).

The first step of performance evaluation, determining the performance objectives, must be accomplished in light of the role of the system being evaluated. In this case, the system under evaluation is the LSINET DDBMS as installed on the LSINET. Both are currently in an intermediate stage of development and several efforts are underway to enhance their capabilities. The role of the LSINET can best be considered as that of an educational test bed. Its use is a function of the projects tasked to the students who use it, and its users change form quarter to quarter. The objective of a DDBMS is to allow the user to access several different local data bases spread over a network as if they were one. As currently implemented, the LSINET DDBMS allows a user to access two relational DBMSs. The users of this system are students involved in thesis

work to upgrade its capabilities and to allow any user to access any of the installed DBMSs. These users need the ability to evaluate their modifications to the system and to compare these modifications to a baseline system. In other words, these students need the ability to conduct before and after performance tests to determine to what degree their modifications have affected system performance. For these reasons, the goal of performance monitoring the LSINET DDBMS will center more around determining the system's efficiency (the quantitative measures of the system's capacity such as component utilization and internal delays) and place less emphasis upon the system's effectiveness (the capability to meet given constraints such as throughput, time requirements or minimum workload)(21:14).

Given a general goal of determining the system's efficiency, the problem of selecting specific areas or indices that best determine this efficiency remains. The specific indices chosen for the LSINET DDBMS system are detailed below along with the reasons for selection.

External System Delay. Defined by such terms as query turnaround time or response time, this is a measure of the time elapsed between input of a user request and receipt of response (21:17). This index is valuable since it will provide the user with the ability to measure the overall effect of the proposed modification in terms of a change in overall delay.

Productivity. This index is defined as the volume of information processed by a system in a unit of time (3:B18). Inclusion of this index will allow a user to determine how well the system performs under differing algorithms or workloads and can assist in answering

questions such as where should the data be located and how can it be
distributed.

Responsiveness. Also known as internal system delay, this index is
a measure of the time between input to a specific component and the
appearance of a corresponding output (3:B19). This is valuable
since it allows for measurement of exact values at a component
level, an aid to users who wish to determine the effect of their
modifications on specific system components.

Allocation. Defined as the types and amounts of resources requested
by or allocated to tasks, this index is useful for users wishing to
trace their modifications or queries in terms of resources used.

Utilization. This index is defined as the percentage of time the
system and its resources are in use. It is useful since it will
allow the users to see how their modifications improve or degrade
component utilization and determine bottlenecks or "heavily"
utilized resources.

Given the goal of determining the system's efficiency in the areas
outlined above, the next step involved the selection of a set of perfor-
mance metrics that encompass the desired indices. At the same time, some
consideration had to be given to the problems of how and when to measure
the selected metrics. Answers to these questions were necessary to
insure that the metrics selected were measurable under the current LSINET
DDBMS system and were meaningful to the method of monitoring selected
(on-line or off-line). Metrics such as number of outstanding queries or
time of last message receipt are of minimal value in an off-line monitor
environment. Additionally, the metrics selected to be monitored in an

31

on-line mode should reflect parameters that can be resolved in a real time mode. For example, say one of the on-line performance metrics is the number of successful access made to an extended data dictionary and the value observed at the time of monitoring was zero. This would probably indicate that there was a problem with the extended directory. Although this is an interesting fact, it is meaningless in a real time environment unless the problem can be corrected in a real time mode. An example of a metric correctable in a real time environment is that of site status. If a site becomes non-operational, this is a useful fact in real time since a number of actions can be undertaken including halting the message traffic to that site and contacting the site to attempt to re-initiate it.

This difference between meaningful and non-meaningful on-line metrics becomes important when the actual metrics selected for monitoring by the LSINET DDBMS monitor are chosen. The determination of whether and where the metric should be monitored was largely a function of the answer to the question, "Is it meaningful in this environment and required by the objectives". Due to the limited system resources, on-line monitoring was restricted to metrics that were very meaningful in a real time environment. All others were deferred to indepth, off-line monitoring. The selected metrics along with a description, method of presentation and reference (if applicable) are given in Table II.

32

TABLE II

Selected Performance Metrics

METRIC	DESCRIPTION	REF	MODE
active sites	the sites currently active on the network, this metric can include all possible sites or just DDBMS sites	None	Online
active process	the number of DDBMS processes active at any given time on the network	(12)	Online
network thruput	total and average amount of network traffic	(2)	Online Offline
DDBMS thruput	total and average amount of traffic generated by the DDBMS	(3)	Online Offline
network delay	time beginning when a message is submitted to the network and ending when it is received by the destination	(22)	Offline
DDBMS response time	time beginning when a querry is submitted and ending when a response is received	(3)	Offline
DDBMS Dictionary Usage	the number of times a dictionary was accessed during an online session, including the success rate of the access	(12)	Offline
DDBMS Dictionary Updates	the number and frequency of dictionary updates during the online session	(3)	Offline

TABLE II

Selected Performance Metrics (Continued)

METRIC	DESCRIPTION	REF	MODE
Network Traffic Distribution	The total traffic flow per source destination link	(2)	Offline
DDBMS Traffic Distribution	The total DDBMS traffic flow per source destination link	(2)	Offline
DDBMS Message Distribution	The distribution of the message by the type received	None	Offline
DDBMS Loading Factor	The ratio of the number of messages generated by a node to the total number of messages generated	(22)	Offline
Number of Packets Queued	The number of packets received by the monitor and awaiting processing	(12)	Online
Average Packet Service Time	The time required to analyze a single packet	(22)	Online

Summary of Chapter Two

This chapter presented the requirements analysis of the stated problem. General concepts in the area of performance measurement and evaluation were explained and a development methodology proposed. This methodology was then applied to the problem and the questions concerning the goals of the performance evaluation and what to measure were answered. Finally the performance metrics to be included in the actual monitor and used in the detailed design stage were defined.

III. System Design

Introduction

This chapter presents the systems design process conducted for the
LSINET DDBMS performance monitor. The requirements and objectives pre-
sented in chapter two form the basis of the design process. The design
process was accomplished in two phases, preliminary system design and
detailed system design. The preliminary system design phase consisted of a
functional requirements analysis documented with Structured Analysis and
Design Technique (SADT) diagrams. The detailed system design phase
included design of program structure and flow. This phase is documented
using structure charts. Both the SADTs and structure charts along with
their associated data directories were prepared in accordance with estab-
lished AFIT standards. Those readers unfamiliar with these tools are
referred to Appendix C, Documentation Tools and Techniques. Prior to pre-
senting an overview of each design phase, this chapter describes the NETOS
encapsulation process upon which both design phases depend for message for-
mats. This chapter concludes with a discussion of the design level the
test plan and a discussion of the design implementation constraints in the
actual LSINET DDBMS monitor.

NETOS Encapsulation Process

The NETOS encapsulation process was designed to conform to the seven-
layer Open Systems Interconnection reference model developed by the Inter-
national Standards Organization (22:15-17). The process begins at the
application layer, (ISO Layer 7) with the creation of a information block
to be transmitted. This information block can be a string, buffer or
entire file, and originate from either the network or LSINET DDBMS system.

36

If the information block to be transmitted is of type LSINET DDBMS, the DDBMS header which includes source and destination identification, unique process identification, a message type, and a time stamp showing time the message was created is appended to the beginning of the block and the block together with the source and destination nodes are transmitted to the presentation layer, (ISO Layer 6).

At the presentation layer, the information block is reviewed and if a file is to be transmitted, the file size is calculated and a pointer to the file is created. The information block (complete with DDBMS header if applicable), the file size, and the source and destination nodes are then transmitted to the session layer (ISO Layer 5). Here the source and destination nodes are transformed into their NETOS equivalent and a source and destination process number determined. The information block is divided into fixed size data buffers. The number of buffers required to transmit the complete message is calculated and a count of the number of buffers currently transmitted is maintained. The total and current buffer counts are appended as the layer five header to the data buffers and the entire buffer as well as the NETOS source, source process number, destination and destination process number are then passed to the transport layer (ISO Layer 4).

At layer 4, the data buffers are divided into messages. The layer 4 header consisting of the source, source process number, destination, destination process number, sequence number (which indicates the sequence number of the message in the data buffer) and use (set to one to indicate the message originated at the transport layer) is affixed to the beginning of the message and the entire message passed to the network layer along with

37

source and destination (duplicating the two fields within the layer 4 header). At layer 3 the source and destination along with a use field are attached to the beginning of the message and a time stamp appended to the end to form a packet. Additionally, the destination is used to determine the port identification. The port identification and the packet are then transferred to the data link layer (ISO Layer 4). Here, a start of text character is attached to the beginning of the packet, a check sum calculated and appended to the end to form the frame. This frame is sent along with the port identification to the physical layer (ISO Layer 1) which transmits it across the network. This process is shown in Figure 7.

Preliminary Design Phase

The first level LSINET DDBMS Performance Monitor SADT, shown in Figure 8, was developed directly from the requirement for both an on-line and off-line monitor detailed in the previous chapter. The "control monitor" activity is responsible for any initialization required as well as for the program flow. Two types of monitor activities are desired. The "conduct on-line analysis" activity has as inputs the message packets from the LSINET. These packets are analyzed and the results of this analysis displayed as performance reports. Additional outputs include CRT messages informing the user of any error conditions and an updated message file containing a copy of all of the message traffic received. The "conduct off-line analysis" activity uses this updated message file to conduct in-depth performance analysis once again producing as output performance reports or CRT error messages.

38
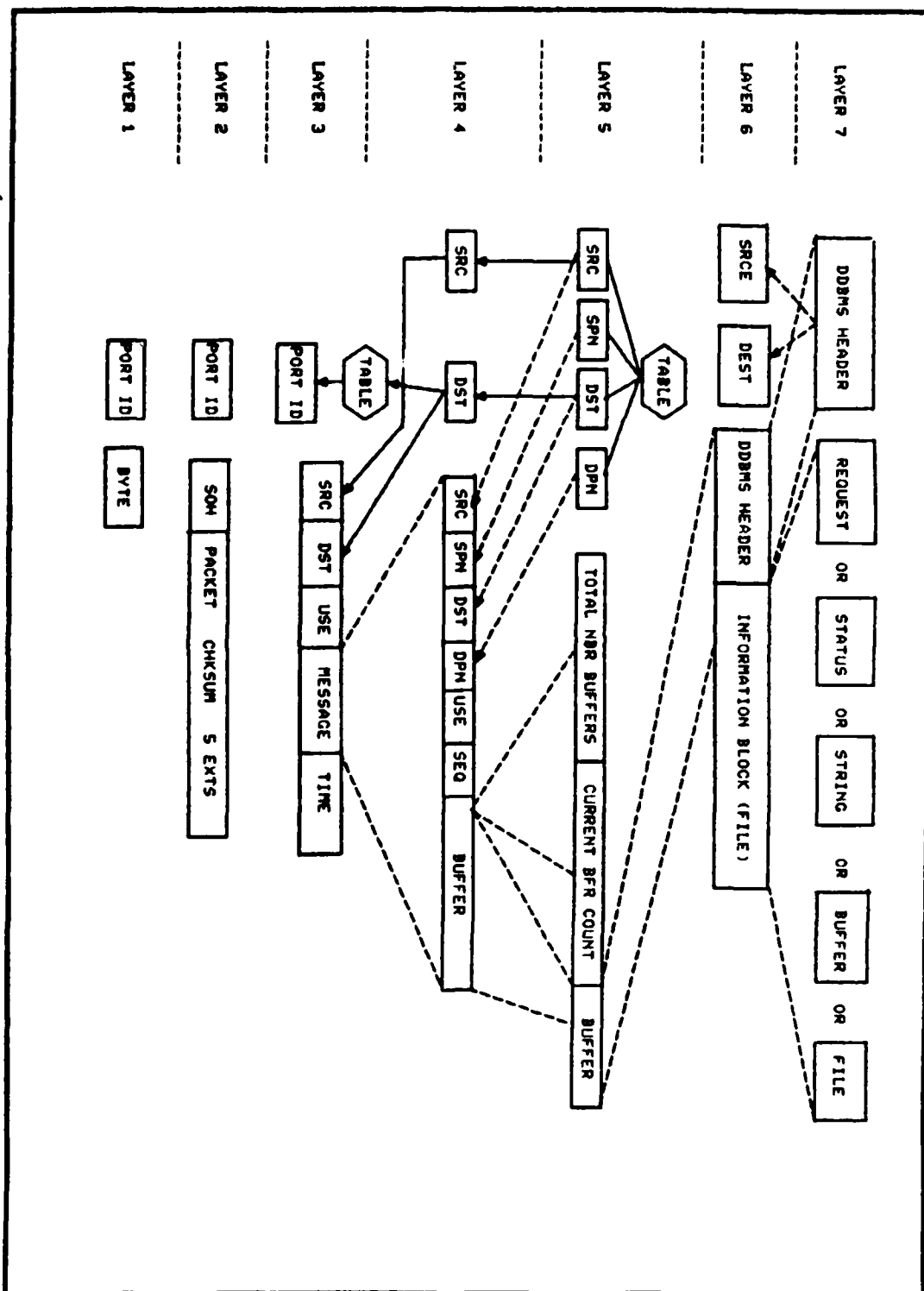
FIGURE 7. NETOS MESSAGE ENCAPSULATION PROCESS

FIGURE 8. DDBMS NETWORK MONITOR SADT

| NODE: A0 | TITLE: DDBMS NETWORK MONITOR | NUMBER: C2 |

40
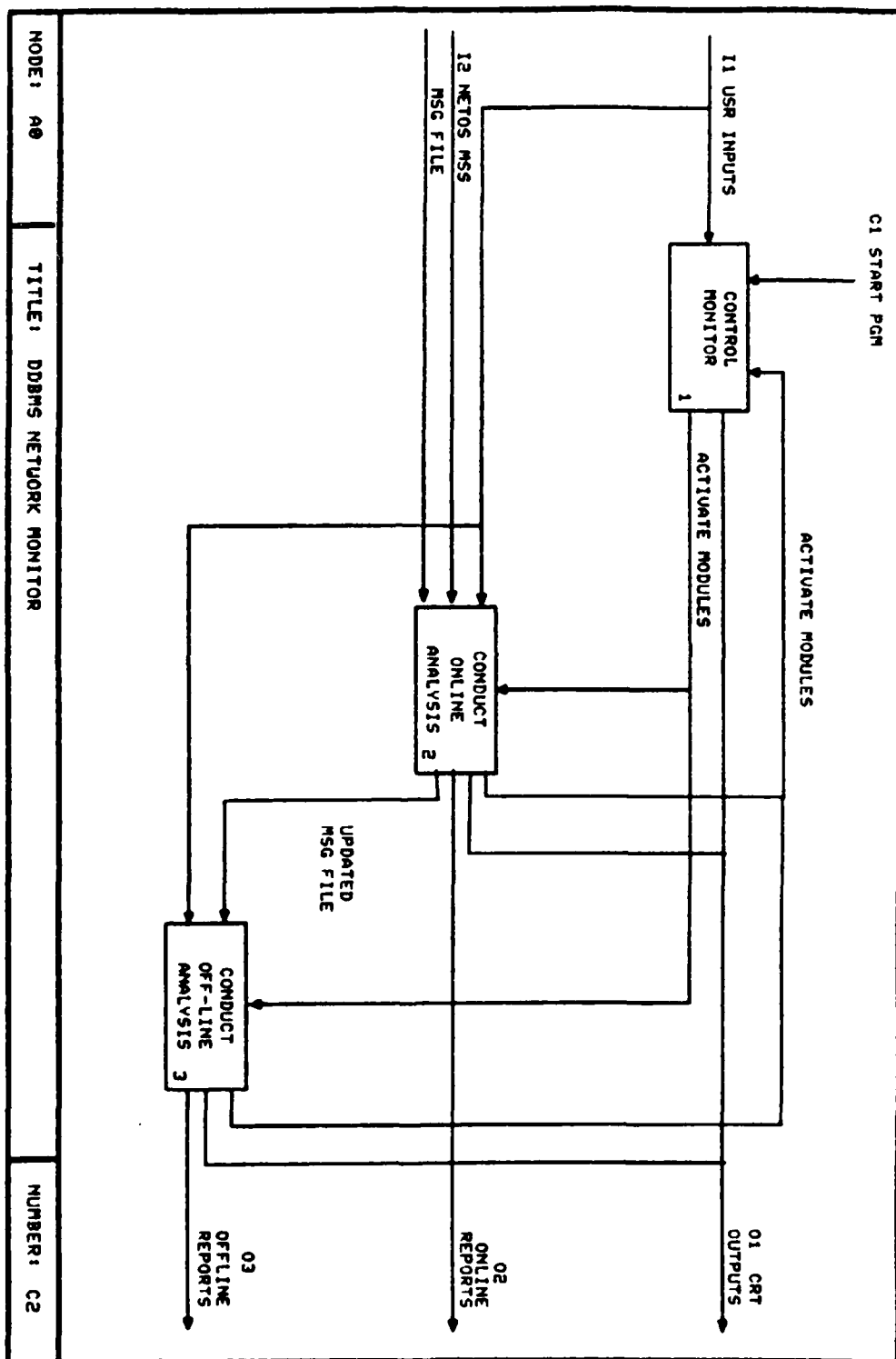
The second level consists of two SADTs, one for each of the analysis activities described above. The first, the "conduct on-line analysis" Activity, shown in Figure 9, decomposes into four separate activities: "available packet", "transfer packet", "analyze message", and "display on-line results". Each of these activities is described below:

Available Packet. This activity monitors the network for the receipt of a message packet. When the message is received, it stores the message in the packet storage area and signals the receipt of a packet to activate the next activity.

Transfer Packet. This activity is responsible for storing the incoming packets in a temporary storage area known as the "tempq". Each packet is sorted into records according to its source host and process number. When enough packets have been received to form a complete record ready for analysis, this activity moves the entry to a permanent storage area known as the "permq" and signals that an entry is ready for analysis. If enough packets are not yet available, control returns to the "available packet" activity. Finally this activity is responsible for updating the message file with a copy of the received packet.

Analyze Packet. This activity is responsible for analyzing the data present in the "permq" entry, modifying the previously calculated performance metrics based on the new information, updating the results records and then signaling when analysis is completed.
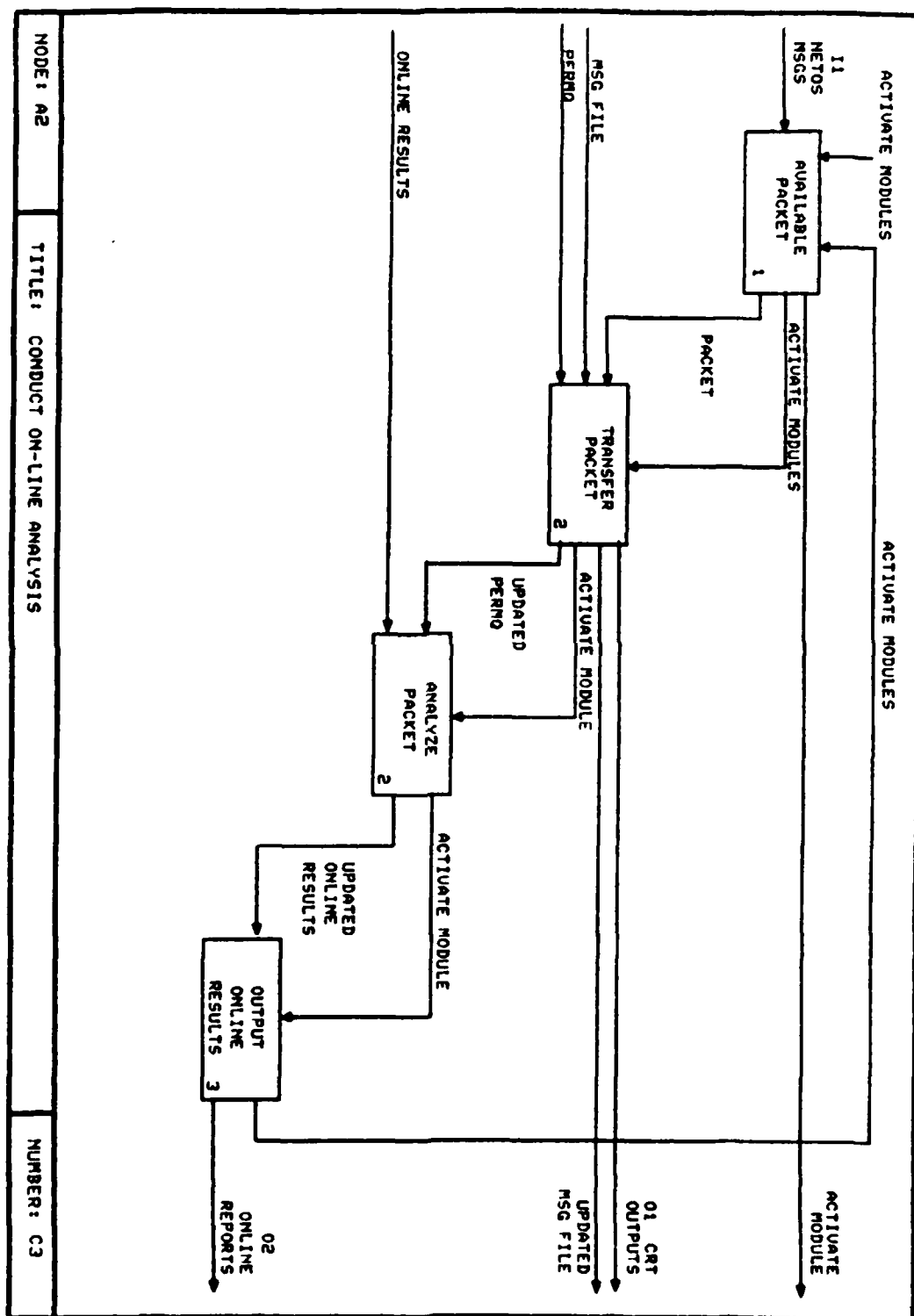
FIGURE 9. CONDUCT ON-LINE ANALYSIS ACTIVITY

NODE: A2   TITLE: CONDUCT ON-LINE ANALYSIS   NUMBER: C3

41

Display Results. This activity updates the on-line performance report screens displayed for the user.

Similarly, the "conduct off-line analysis" activity shown in Figure 10 can be broken down into the activities described below:

Transform Message File. This activity is responsible for taking inputs from the message file and presenting them to the "conduct off -line analysis" activity in a form designed to aid analysis. (An example might be the reconstructing of the DDBMS packets into partial messages for analysis.) This transformation should take place without destroying the original message file and control should pass to the "conduct off-line analysis" activity whenever a complete entry is available for analysis.

Analyze Off-line Messages. This activity is responsible for analyzing the data present in the file entries provided by the "transform message file" activity. The selection of the actual performance metrics analyzed is accomplished via user inputs. Upon complete analysis of the message file, this activity signals the completion of the analysis process.

Display Off-line Reports. This activity generates a comprehensive set of output reports upon completion of the analysis process.

For readers desiring more detail on the preliminary systems design process, a complete design package including the third level SADTs and data directory entries for each of the activities and data elements is provided in Appendix E, Preliminary Systems Design Documentation.

Detailed Design Phase

Upon completion of the preliminary design phase, the detailed design phase commenced. This phase was responsible for detailing program structure and flow of control. This was accomplished using structure

NODE: A3   TITLE: CONDUCT OFF-LINE ANALYSIS   NUMBER: C6
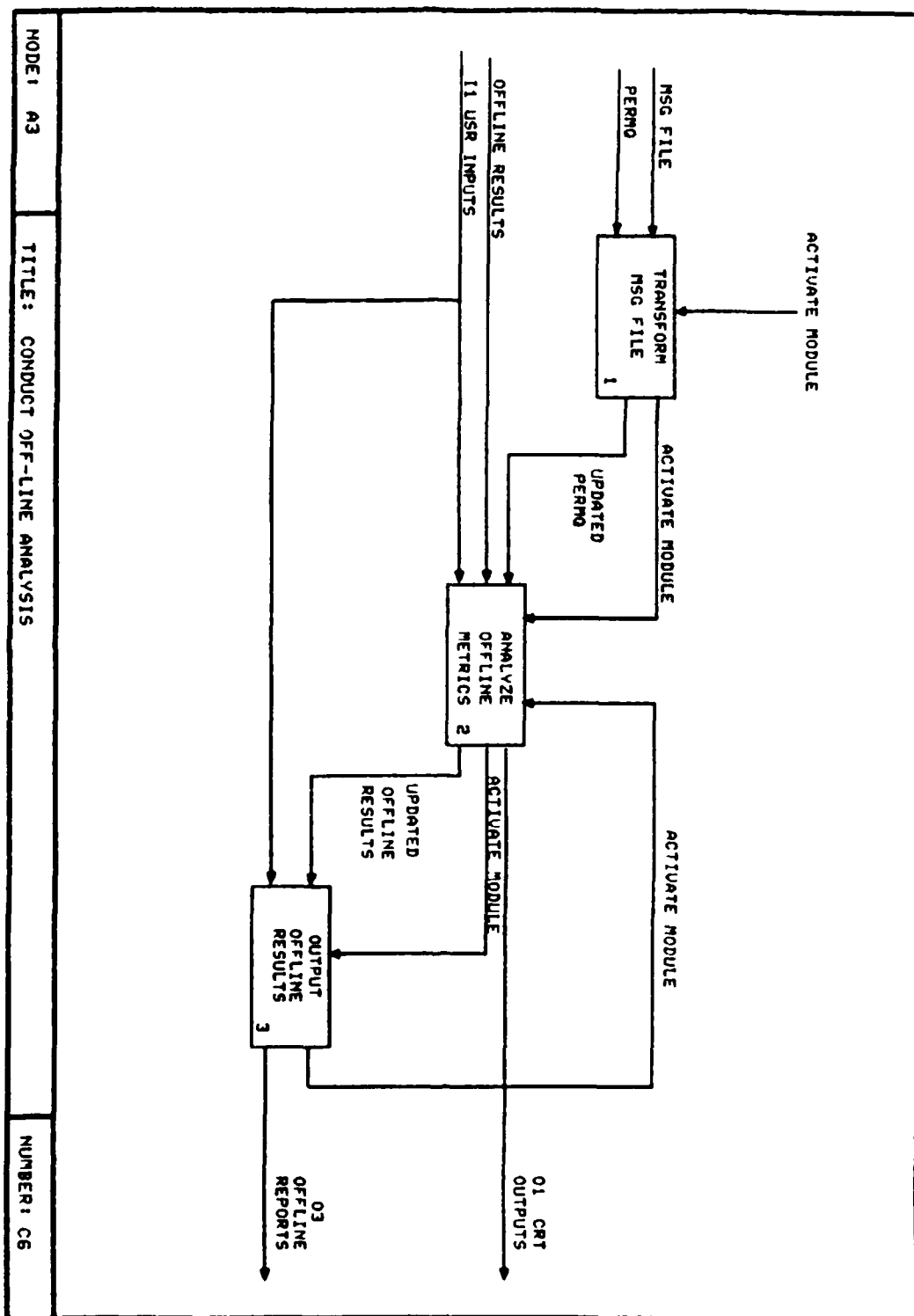
FIGURE 10.   CONDUCT OFF-LINE ANALYSIS ACTIVITY

43

charts to enhance and clarify the SADTs developed earlier. The high level

structure charts for the "on-line" and "off-line" processes are described

below with differences between the structure charts and their associated

SADTs noted. A complete set of structure charts and associated data

directory entries is provided in Appendix F, Detailed Systems Design Docu-

mentation.

The first level structure chart shown in Figure 11 is directly derived

from the first level SADT (Figure 8). In this structure, program control

flows between the "control monitor" process and the "on-line analysis" and

"off-line analysis" processes with process selection determined by user

input. Inter-dependencies between second level processes are kept to a

minimum with the only shared parameter being the message file created by

the "on-line analysis" process and used by the "off-line analysis" process.

The second level structure charts detail the operation of each of the

analysis processes. The first, the "on-line analysis" process, is shown is

Figure 12. It differs somewhat from its associated second level SADT

(Figure 9), reflecting the need for initialization and error detection not

specifically covered in the SADT. This process decomposes into the

following third level processes:

Init Qryscn. This process is responsible for querying the users for

the on-line performance metrics to be displayed by the real time monitor.

This process, not envisioned during the preliminary design phase, resulted

from the decision to allow the user to tailor the monitor to the largest

extent possible, displaying the specific performance results that meet the

user's needs.
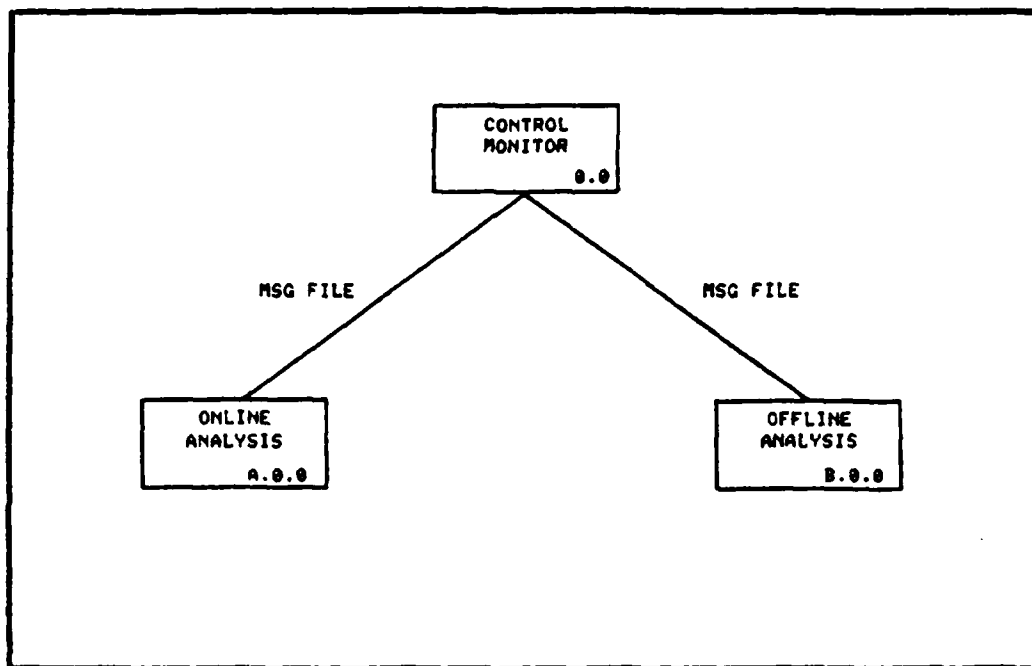
FIGURE 11. PERFORMANCE MONITOR STRUCTURE CHART
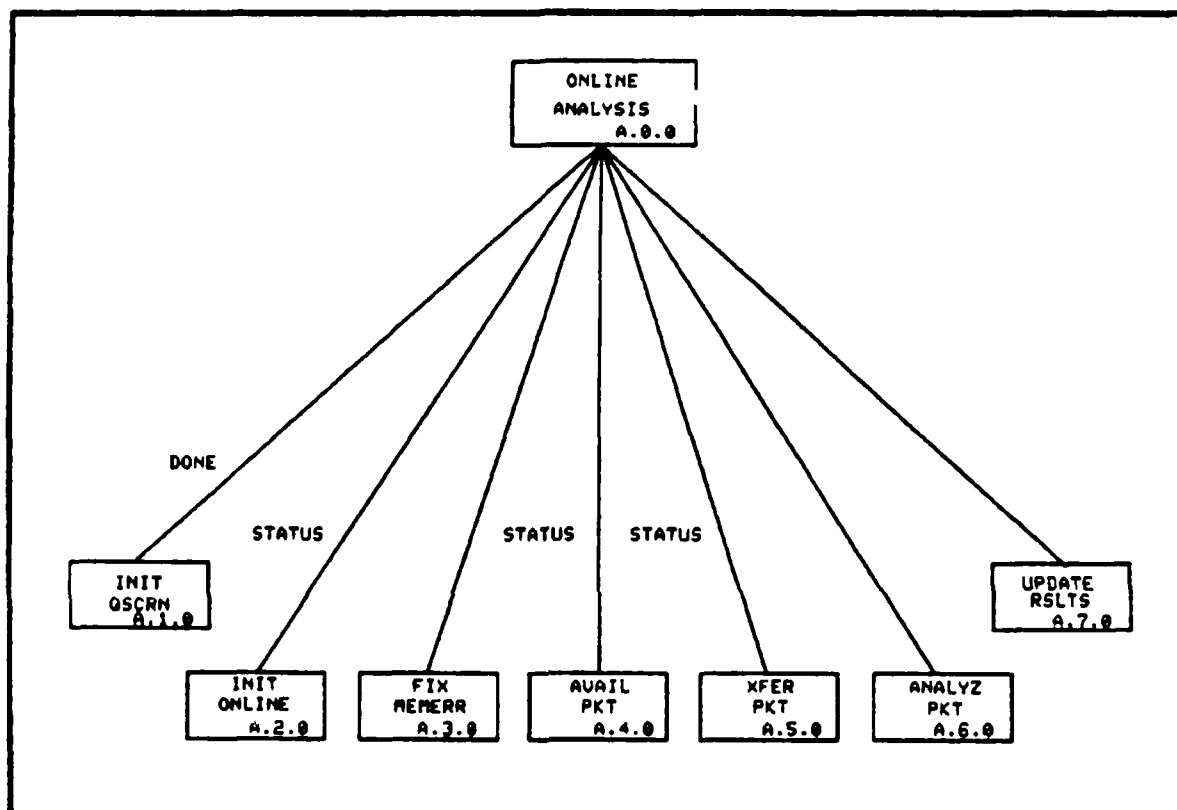
FIGURE 12. ONLINE ANALYSIS STRUCTURE CHART

Init On-line. This process provides initialization for any required global values and structures and was originally included as part of the control monitor functions in the preliminary design phase. The decision to move it resulted from the decision to minimize the inter-dependencies between the "on-line analysis" and "off-line analysis" processes.

Fix Memerr. Although not included in the preliminary design SADTs, the on-line monitor requires the ability to continue operations when faced with correctable errors (such as an out of memory condition). This process is responsible for providing this ability, re-initializing the monitor while minimizing the loss of data and insuring the result's integrity.

Avail Packet. This process corresponds directly with the "avail packet" activity described above. A status signal is returned to the calling process (on-line analysis), informing that process when a message packet is available.

Xfer Packet. Corresponding to the "transfer packet" activity, this process is responsible for storing the packet into the appropriate record, (either "tempq" or "permq") and signaling the calling process (on-line analysis) via a status flag when a complete entry is ready for analysis.

Analyze Pkt. This process is responsible for performing the on-line analysis on the completed entries and updating the results statistics as necessary. The process corresponds directly to the "analyze message" activity in the SADTs.

Update Rslts. This process, corresponding to the "display results" activity in the SADTs, is responsible for structuring and displaying the on-line results.

The second layer structure chart for the "off-line analysis: process as shown in Figure 13, differs drastically from its related SADT (Figure 10). This difference reflects the decision to keep the "off-line analysis" process as independent and modularized as possible. Reasons for this decision are covered in the section that discusses the use of the design in the actual LSINET DDBMS monitor. The result of this decision however, is the segregation of each of the performance metrics with the activities presented in the SADT implied in each of the second layer processes. Each of the processes is described below:

Det Thru Put. This process is responsible for querying the user for the type of thruput to be measured (Network or DDBMS) and the type of results display desired. Based upon the replies to these queries, analysis of the message file is conducted, and the resulting statistics displayed for the user in the desired format.

Det Resp Time. This process is responsible for analyzing the message file contents to determine the response times for the selected performance metric. Response time can be determined for Network or DDBMS traffic, and on a packet or message level.

Det Dict Access. This process is responsible for conducting the analysis of the message file that determines the access statistics for the DDBMS central, local, and extended Data Directories.

Design Test Plan

The design level test plan developed in conjunction with the Detailed Design Phase, served three purposes. First, it insured that all of the possible inputs to each process was discovered and their results anticipated. Second, it served as a guide to the actual software implementation

48
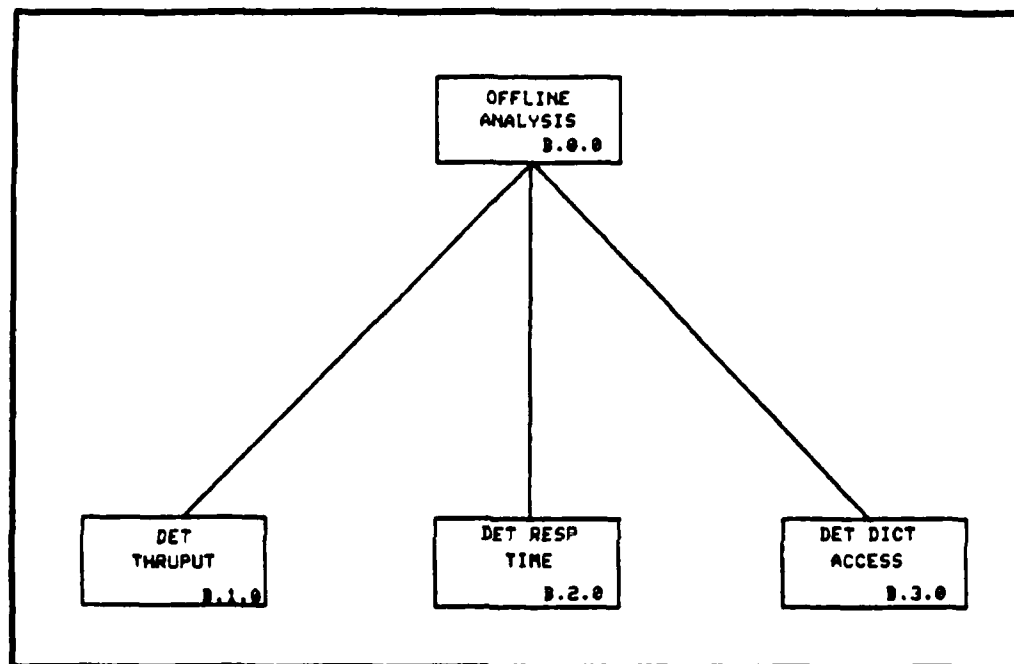
FIGURE 13.   OFF-LINE ANALYSIS STRUCTURE CHART

allowing for greater program clarity and completeness. Third, it was used as a major portion of the module level test plan required in the implementation and test phase. This design test plan is written with the overall intent to test the validity of the design (i.e. to find errors in the design) and so is extremely detailed. When incorporated in the implementation phase test plan, it will serve to accomplish module level correctness testing. This means that each module will be tested in isolation before integrated with the rest of the modules. (Integration testing of the modules is described in detail in the next chapter.) A sample of the design test plan is shown in Table 3, with the complete plan presented in Appendix B, Test Documentation.

Design Implementation

The complete detailed design package included in Appendix F shows the processes required in implementing the DDBMS performance monitor. To the largest extent possible, the design phase was conducted independent of machine implementation and for that reason some of the design processes will require restructuring to fit the actual monitor environment. Details on this restructuring are provided in the next chapter.

The knowledge that this monitor would be running on a minicomputer with limited amounts of storage space, did effect some portions of the design package. It reduced the performance metrics monitored in the on-line session to those that can be analyzed quickly and reacted to in a real time environment. It also caused the off-line monitor to be designed so that each set of sub-modules was independent and could be implemented as a set of small programs rather than one large one. This design does not

TABLE III

Sample Design Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Control Monitor (0.0)	Central system activated and passing messages	Run Program	Query user for next action	
		Action = run online	Call online analysis module	
		Action = run offline	Call offline analysis module	
		Action = exit	Exit program	
Online Analysis (A0.0)	Central system activated and passing messages	Call module	Call initialization modules	
			Query user for output format	
			Print message and Poll for next. pkt	
		Status flag = PKTNAVL	Print message and Poll for next pkt	
		Status flag = PKTAVL	Call xfer_pkt module	
		Status flag = HDRAVL	Call anlyz_pkt module	
		Status flag = HDRNAVL	Poll for next pkt	
		Status flag = MEMERR	Call fix_memerr module	
		Status flag = IRRERR	Print message and terminate program	

51

effect the monitors ability to be implemented on a large scale system. The monitor has simply been designed to run more efficiently on a small system.

Summary of Chapter Three

This chapter discussed the two phase system design process conducted for the LSINET DDBMS. The processes and activities involved in each phases were described in detail and their interfaces outlined. Additionally a design level test plan was included and the approach for using the design in the implementation discussed.

IV. Implementation and Testing

Introduction

This chapter discusses the methods used to implement and test the LSINET DDBMS performance monitor. First, the details the exact monitor environment are discussed, specifically the hardware and software selected, network topology, and type of center. Next, descriptions of the actual programs used to implement the monitor are provided. Although the performance monitor was originally envisioned as a single program with two phases, on-line analysis and off-line analysis, this proved difficult to implement due to the size constraints of the LSI-11 micro-computers. Instead, it was implemented as two separate programs. Each of these programs are described fully, including the performance metrics calculated, module descriptions, input and output screen formats, and all assumptions made. Finally the complete test plan for the monitor is described, along with the results of the test.

Monitor Environment

As mentioned previously the performance monitor was implemented on the LSINET located in the Digital Equipment Laboratory. This network has a star topology with one node, the central site, responsible for message passing. (A complete description of the LSINET and the NETOS communications protocol is available in Appendix A, Systems Environment.) Unlike most centralized computing systems, the LSINET contains no software reporting tools or cost accounting packages that could be used to assist in performance analysis. This lack of existing performance monitoring tools, coupled with the type of network topology, caused the decision to implement a passive (no traffic generated), hybrid (specialized software

53

programs installed on dedicated hardware) monitor at a centralized (single) site. This minimized the artifact introduced in the existing communications software and the only change required was to send a copy of all message traffic to the monitoring center. In a ring or bus topology, no artifact would have been introduced since in a single source media, the monitoring center can be configured to be a universal receiver and a separate copy of each transmitted message is not required.

The actual monitoring center was implemented on an dedicated LSINET workstation. At first the workstation selected to host the monitor was an Intel System 310. Difficulties with this device, (detailed in Appendix D) caused the implementation of the monitor on an operational LSI-11 workstation instead. No specific workstation is required, but the central site software only recognizes three nodes and monitor hosts, systems A, S and K. In keeping with the existing software on the network, all program development was accomplished in Whitesmith's C.

On-line Monitor Design and Implementation

Performance Metrics Calculated. The following is a description of the performance metrics which are monitored in the on-line session, how they are obtained and where they are calculated in the on-line program.

A. Total count of network packets - This is a simple count of all the packets received by the monitor regardless of packet type and it is calculated as a simple increment of the value of tot_npkt in the main module.

B. Total count of DDBMS packets - Also a simple count, the tot_ddpkt metric keeps track of all the DDBMS packets received and is calculated in the xfer_pkt module.

54

C. Total processes - Slightly more complicated, the tot_prc metric

is a count of the total number of unique DDBMS queries received.

It is calculated by monitoring the DDBMS packet for a message type

of "BEG". When this is received, the process number is noted and

the tot_prc count incremented. The analysis of this metric occurs

in the anlyz_pkt module.

D. Active processes - The metric atv_prc is a count of the total

number of currently active processes. Calculated in the anlyz_pkt

module, this count is incremented when a DDBMS packet of type "BEG"

is received and decremented when type "STP" is received.

E. Active sites - This metric is calculated from the values in the

field sarray (site array), a 26 character array that reflects the

site status for each of the 26 allowable sites. A one in any

position means the site is active, a zero means it is inactive.

Initially calculated by interpreting the entries in a DDBMS packet

of type "DRC", the atv_site metric is updated with the receipt of

"ASM" and "DSM' type messages. This metric is calculated in the

anlyz_pkt by simply counting the number of ones entered in the

sarray field.

E. Inactive sites - The opposite of atv_site, this metric is

calculated during the update_rslts module by subtracting the value

in atv_site from the total number of allowable sites.

F. Arrival Rates - Arrival rates for the network traffic, DDBMS

traffic and processes are calculated by dividing the total count of

packets received (or processes started) by the length of time the

monitor has been running. Calculated during the anlyz_pkt module,

55

these metrics are converted to messages per minute for output.

G. Time last packet received - Calculated during the main module, the metric is simply a time stamp showing when the last module was received. It is printed to the screen when the monitor is receives a packet for analysis.

H. Number of packets queued - The lvl_2q metric is a count of the number of packets queued at the ISO 2 level awaiting processing. Designed somewhat as a semaphore, this metric is incremented when ISO 2 level receives a packet and decremented when the module avail_pkt removes a packet for analysis.

Module Design. The complete list of the modules incorporated in the on-line performance monitor can be found in the file header block. This list includes a brief description of each module. Additionally, the module header block contains a more detailed description. Code for all of the modules can be found in Appendix G, On-line Program Documentation. For this reason most of the modules are not described again here. However, because simpler design alternatives exist for a few of the on-line modules, the rational used for selecting the module design used is included for clarity. The modules are:

A. Xfer_pkt - This is a complicated module required because to insure maximum network flexibility, the program does not assume that the size of the DDBMS header is less than the size of the transmitted data packet. This means that there are two alternatives in dealing with the problem of split messages, total message reconstruction or partial message reconstruction. Total message reconstruction was rejected because the messages could have been

56

files which would require a prohibitive amount of storage to reconstruct. This means that messages must be partially reconstructed until a complete DDBMS header is formed and then parsed into appropriate fields for analysis. (Total message reconstruction is not a requirement.) This was accomplished using two queues, the "tempq" where packets are stored until a complete DDBMS header is available and the "permq" which contains the parsed fields of the DDBMS message. To determine if a packet is required to form the DDBMS header, the block count and sequence number are checked. In the initialization process, the number of blocks required to form a DDBMS header was calculated. The block count of the current message is compared to the block count required and passed to the xfer_queue module if the packet is a required part of the header. If not the received packet count is incremented to reflect the arrival of the packet, a check is made to see if all the outstanding packets of the message have been received and the proper signals generated (either complete message available or resume polling).

B. Xfer_queue - This module is responsible for reconstructing enough of the received message to form a DDBMS header. It is complicated by the fact that the assumptions were made that packets would not necessarily be arriving in order. This means that packet 2 from host A may arrive before packet 1 from host A or that packets from hosts A and B may arrive inter-mingled. A further assumption was made that all of the packets forming a complete block would arrive before the next block of a message was transmitted. In order

57

to determine to which message a packet belongs, a compare array
(cmpary) has been formed from the network header and is assumed
unique for each transmitted message. This array is then compared to
each record in the "tempq" to find the message the packet belongs
to. Once found, the block count and sequence number are used to
calculated where in the record the information field of the packet
should be stored. The record is then checked for completeness and
the calling module appropriately signaled. An alternative design was
to the partial parsing of packets into the "tempq" was to link the
pointers to the packets together until a complete packet was formed,
and then parsing that packet into the "permq" as required. This
method was attempted but problems with correctly linking packets
together and overflowing the storage capabilities of the
microprocessors caused the simpler, albeit more time consuming
process described above to be implemented.

C. Search_mtab - This module is used to convert a three character
DDBMS message type into a unique integer equivalent. This is done
by searching the message table for the message type and returning
the message integer. The message table does not contain all of the
allowable DDBMS message types, rather just the ones needed by the
monitor to calculate performance metrics. If a message type is not
found the module returns a zero. This method of table build and
look up was implemented instead of setting up static arrays in the
actual programs because it allowed the most flexibility in actual
implementation.

```
Welcome to the DDBMS Network Monitor Program

This program has been designed to perform as an online monitor
    for the DEL DDBMS implemented on the LSINET.


You have a choice of the following actions:

        A ===> Initiate/Continue Online Monitor
                    and display DDBMS Site information
        B ===> Initiate/Continue Online Monitor
                    and display Primary Metrics Screen
        C ===> Change Floppy Disks

        X ===> Exit

        ENTER YOUR CHOICE NOW ===>
```

FIGURE 14. ONLINE MONITOR INPUT SCREEN

D. Fix_memerr - If the processor runs out of storage space during execution, this module tries to fix the problem and resume operation. Rather than freeing all allocated storage, this module first de-allocates all the storage allocated to the "permq" and the "tempq". If enough storage space has not been freed, the packet queue is then de-allocated after first storing the packets into the off-line file. In this way, the chances of completely losing message traffic are reduced. In either event, a warning message prints to the screen to alert the user of the situation.

User Interface. The on-line monitor has one input and two output screens. The input screen, shown in Figure 14, prints a menu of the actions available to the user of the monitor. The alternative to a menu driven on-line monitor would have been having the user enter a command without the menu prompt. Since familiarity with the monitor could not be assumed, a help screen would have had to be implemented to give the user the available options. For this reason, it was decided to implement the a menu driven monitor. The output screens were designed to maximize clarity while minimizing clutter. For this reason the output was divided into two screens with one showing the status of the sites and the other the arrival rates and packet counts. The first output screen, Figure 15, presents the overall results of the analysis, showing received packet counts for DDBMS and Network packets as well as their respective arrival rates. Counts of the total number of active processes, the total number of processes received, the total number of active and inactive sites and the process arrival rates are also included. Finally, three time stamps showing current time, the time analysis began, and the time the last

```
                    DDBMS      ONLINE     MONITOR


NBR ACTIVE SITES:          NBR INACTIVE SITES:        NBR QUEUED PKTS:


TRAFFIC INFO:
     Total pkts received:                   Avg Arrival Rate:
     Total DDBMS pkts received:             Avg Arrival Rate:
     DDBMS Percentage of Total:


PROCESS INFO:
     Total Nbr Processes rcvd:              Avg Arrival Rate:
     Nbr Active Processes:

START TIME:
TIME LAST PACKET RECEIVED:
CURRENT TIME:



*****************************************************************
```

FIGURE 15. ONLINE MONITOR PRIMARY OUTPUT SCREEN

packet was received are also included. At the users discretion, the secondary output screen, Figure 16, showing the active and inactive DDBMS sites can be displayed. The lower three lines of both output screens are reserved for error and warning messages from the monitor.

Program Assumptions. Several assumptions were made in the course of coding and implementing the on-line monitor program. These assumptions are listed below:

A. Save file - All message traffic is saved to the DDMON.MSG file in its received format. This file is assumed resident on the floppy disk located in drive DX1. Since the C compiler does not support the C file "appends" option, this file is opened at the beginning of execution and remains open until the program terminates. This means that no tests for an "out of disk space" condition can be conducted and should a file overflow occur, the program will abort to the operating system. To remedy this problem, the size of the DDMON.MSG file was fixed at 1600 blocks of 140 characters, roughly half the space on a floppy disk. The program counts the number of packets sent to the disk and when that count reaches 1600, the file is closed and the user signaled with a warning message that packet traffic is no longer being saved. Until a new disk is loaded and the save function restarted, all received packets are analyzed and discarded.

B. The output screens are painted once and then specific positions updated as required. For this reason the lower three lines of the screen are used for all monitor messages to the user.

```
                    DDBMS SITE STATUS SCREEN


         ACTIVE SITES                          INACTIVE SITES










              TOTAL:                          TOTAL:

          CURRENT TIME:



     **********************************************************************
```

FIGURE 16. ONLINE MONITOR SECONDARY OUTPUT SCREEN

C. The monitor will run without interruption until the user strikes a key. This action will cause the program to return to the input screen menu where the user can choose a different option. This does not cause a reset of the on-line monitor and statistics are calculated from the time the monitor was first started.

D. The monitor has "hooks" built in to allow monitoring of its operation as well. These hooks are activated by setting the constant MONIT to a one and recompiling and linking the program. In this version of the monitor, module entry and exit will cause messages and time stamps showing when these actions occurred to be printed to the file MONTR.DAT.

E. All packets can be uniquely identified as DDBMS type packets by the source and source process number values located in the network header and contained in each transmitted packet. The packet is assumed to be a DDBMS type packet if its source and source process number are of type DDBMS (this is determined by the values entered in the DDTAB.DAT file, loaded from floppy disk at program execution). A packet generated by a DDBMS host and process cannot be transmitted to a non-DDBMS host and process and vice versa.

F. It is assumed that all the packets constituting a message from one site and process will be received before the next message from that site is begun, although the packets do not have to be received in order. Secondly it is assumed that a message can be reconstructed from the information contained in the network packet headers and access to the DDBMS header information is not required for reconstruction.

Off-line Monitor Design and Implementation

Performance Metrics Calculated - The following is a description of the performance metrics monitored for in the off-line monitor. Also included is a discussion of how the metrics are obtained from the traffic and where they are calculated in the monitor program.

A. Traffic Distribution - This metric calculates amount of packet traffic passed between a source-destination pair for the entire monitor period. This metric can be calculated for all the transferred packets or just the DDBMS type packets. Either way, this metric is a simple count of the the transmitted packets sorted by source and destination and calculated in the net_site module.

B. Total Traffic Generated - This metric calculates the number of packets each source generated during the on-line session, regardless of destination. Also calculated in the net_site module, this metric can be calculated for all packet traffic or just DDBMS packets.

C. Total Traffic Received - This metric calculates the number of packets each destination received during the on-line session, regardless of source. As above, the metric is calculated during the net_site module and can be tailored to monitor all traffic or just DDBMS traffic.

D. Total Packets Monitored - Calculated by the net_site module, this metric is a simple count of the number of packets monitored either network or DDBMS.

E. Time Distribution - This metric calculates the packet traffic, either network or DDBMS, by time generated. It is calculated by the

net_time module and is a simple count of the packets by the hour
they were generated by the ISO level three function.

G. DDBMS Message Distribution - This metric is calculated by the
dd_msg module and calculates the traffic distribution for DDBMS
messages, not packets, by message type.

H. Process Traffic Generated - This metric is calculated in two
forms. In the prcss_pkt_traf module the metric counts the number of
packets generated by each unique process. This is sorted by source
and source process number. In prcss_pkt_traf, the metric is
calculated for the messages generated for each unique DDBMS process.

I. DDBMS Response Time - Calculated in the det_ddresp module, this
metric measured how long it took to satisfy each query type.

J. Directory Utilization - This metric is a measure of the number
of hits and finds for each LNDD, ENDD, and CNDD and is calculated by
the det_dict_accss module.

Module Design. A list of the modules contained in the off-line
monitor program is given in the file header along with a brief descrip-
tion of each module. Detailed descriptions are included in the module
headers. Several modules designed for the on-line monitor are also
included in the off-line monitor. When required, the methods of recon-
structing partial DDBMS messages to get complete DDBMS headers are the
same as those used in the on-line monitor.   With the exception of two
modules, the off-line metrics are calculated by selecting the metric to
be analyzed and then searching the message save file for information per-
taining to that metric. In this way, the save file is only read once for
each metric calculated. The exceptions to this rule were the modules

66

```
                    DDBMS OFFLINE MONITOR

                   LEVEL TWO -- THRUPUT


Your choice of actions is:

        A ===>  Analyze Network Packet Traffic by Site
        B ===>  Analyze DDBMS Packet Traffic by Site
        C ===>  Analyze Network Packet Traffic by Time Period
        D ===>  Analyze DDBMS Packet Traffic by Time Period
        E ===>  Analyze DDBMS Packet Traffic by MSG Type
        F ===>  Analyze Network Process Traffic by Pkt Generated
        G ===>  Analyze DDBMS Process Traffic by MSG Generated
        X ===>  Exit



ENTER YOUR CHOICE NOW ===>
```

FIGURE 17. OFFLINE MONITOR SAMPLE INPUT SCREEN

det_dict_acss and dd_resp_time. Both of these modules had to be redesigned so that the amount of storage used was not a function of the save file size (See Appendix B for more details). In order to accomplish this, after the metric was selected, the file was first read for the next process number to be analyzed. The complete file was analyzed for information regarding that process number, statistics calculated and the file closed. After one process number was completed, another was selected. The analysis process continued in this fashion until all of the monitored processes were analyzed for the desired metrics.

User Interface. The user interface required for the off-line monitor is more slightly more complicated than that of the on-line monitor. Unlike the on-line monitor, the off-line monitor is menu driven, allowing the user the ability to select a subset of the total available metrics and thus tailor the monitor to meet his/her unique needs. This results in several available input screens, one for each level of detail. A sample input screen is included in Figure 17. The output screens have been designed to meet the size constraints of the screen. A sample output screen is included in Figure 18.

Program Assumptions. The assumptions made for the off-line monitor program are as follows:

   A. Each metric is calculated and displayed by a distinct group of sub modules. This group is by intent independent of the rest of the monitor program and could, if necessary, function alone. For this reason the message save file is declared, opened, read, and closed in several of the modules.

68

```
                    NETWORK DISTRIBUTION TOTALS


    S\D   A    B    C    D    E    F    G    H    I    J    K    L    M

     A    0    0    2    5    0    3    7    8    0    0    0    4    2
     B    0    0    0    0    0    0    0    0    0    0    0    0    0
     C    3    0    0    7    3    6    0    9    0    3    0    0    0
     D    1    0    2    0    2    4    0    7    0    1    0    5    3
     E    4    0    0    0    0    0    6    4    0    0    0    0    2
     F    0    0    1    2    0    0    0    1    0    0    0    0    0
     G    1    0    2    4    5    2    0    1    0    1    0    5    0
     H    3    0    0    0    1    0    0    0    0    1    0    0    1
     I    0    0    0    0    0    0    0    0    0    0    0    0    0
     J    0    0    1    2    6    4    0    8    0    6    0    0    1
     K    0    0    0    0    0    0    0    0    0    0    0    0    0
     L    1    0    6    9    2    0    0    7    0    1    0    1    5
     M    0    0    2    4    9    0    0    1    0    1    0    3    5
```

FIGURE 18.   OFFLINE MONITOR SAMPLE OUTPUT SCREEN

B. As in the on-line version, monitor "hooks" exist which, when activated, allow the monitor's performance to be monitored.

C. The monitor is menu driven, and requires frequent interaction with the user. For this reason, it cannot run completely unattended as the on-line monitor can.   When user intervention is required however, a bell on the terminal sounds to alert the user.

D. An error condition in the off-line monitor will cause a termination of the metric analysis, a printout of the cause of the error (i.e. insufficient storage available) and a return to the next higher level of the off-line menu.

E. As in the on-line monitor, all packets can be uniquely identified as type DDBMS by their source and source process number. Likewise, a packet generated by a DDBMS host and process is cannot be transmitted to a non-DDBMS host and process.

Changes to Existing Programs

One of the goals in designing the program monitor was to minimize the impact on the existing programs. To the largest extent possible this was accomplished. In four areas, the existing programs were found lacking and changes made. These areas are as follows:

Time Stamps. As implemented, the LSINET traffic had no time stamp to indicate when traffic was generated or passed. A time stamp of some sort was required to test for delays and show traffic distribution over a given time period. For this reason, it was decided to modify the existing ISO level software to include time stamps. The next decision was to determine which layers should add the time stamp. There was a valid argument to add a time stamp at each layer of the ISO software. In

70

this way the packets could be monitored to see how much time each layer

required and decisions to optimize the code at given layers could be

made. The problem with this argument proved to be the amount of monitor

overhead added to each packet. Each time stamp requires ten characters,

so adding seven would increase the overhead in each transmitted packet

by 70 characters. Given the current information field is 120 characters

and the overhead for existing ISO layer headers and trailers was 18

characters, an overhead of 88 characters per packet (over 50 percent) was

deemed excessive. The decision was made instead to add time stamps at

the application program level, ISO layer 7, where the information to be

transmitted or requested is first created and then for each packet at the

network level, ISO layer 3. This reduced the overhead to ten characters

for each complete message and ten characters for every packet.

Monitor Site. The NETOS central system had already been modified to

send a copy of all traffic to a set node, system L, where the monitor was

originally scheduled to reside. Due to number of projects requiring the

use of system L and ability to run the monitor program on any of the

available nodes, the was changed to allow the network manager to select

the monitor node at the time the central system is initiated or change it

during operation.

Unique Process Number. A problem discovered in the existing DDBMS

software was the fact that the messages had no identification and once

divided into buffers and packets, reconstruction was impossible if more

than one message was present on the network. Also needed was the ability

to relate reply messages to queries. For these reasons, a unique process

number was assigned to each process. All messages and queries spawned by

the original process were assigned the same process number. This allowed
the traffic to be traced to a specific process.

Buffer/Block Count. At ISO level five, the files or buffers to be
transmitted were divided into blocks and passed to layer 4. Due to the
storage constraints of the monitor hardware, it was decided not to try to
reconstruct each message but rather simply keep track of the total number
of layer 5 blocks or layer 3 packets expected. For this reason, layer 5
software was changed to include a header of two fields, total block count
which is determined by calculating the size of the file to be trans-
mitted, and current block count, the count of the block as it is passed
to layer 4.

Performance Monitor Testing

Basic Definitions. Given that it is impossible to prove a program
is totally free from errors, testing is best defined as the process of
executing a program with the intention of finding errors (18:169).
Testing can be used to both validate and verify software programs. When
a program is tested in a simulated environment with the goal of insuring
that the software being developed corresponds to specification, it is
being verified. When testing is conducted in a real environment to
ensure that the software is meeting its objectives, it is validated (22).
The goal of testing should be to both validate and verify a program.

There are several techniques that have been developed for use in
testing. Three methods commonly used are top-down, bottom-up, and mixed
testing. In top-down testing the modules of a program are tested and
interfaced in a fixed sequence (20:239). First the control module is
written and tested with lower level modules represented by stubs.

Testing of the program continues on a level by level basis, until all levels have been integrated and tested. Bottom-up testing reverses this procedure, developing the bottom most modules first and testing them via program drivers. Upper level programs are then designed and tested and the processes continues until all modules have been tested (20:239). Mixed testing or kernel testing, is an adaptation of the other two types. Critical modules are designed first and tested with drivers or stubs, which ever is appropriate. Then, both preceding and succeeding level modules are added and tested. This process continued until all modules are tested.

In this effort, most of the testing accomplished was concerned with verifying the developed software and ensuring that it worked according to specification. Due to unavoidable problems, no actual DDBMS traffic existed to accomplish the validation testing. Instead artificial traffic was sent across the network and some validation testing was accomplished. Testing was conducted in four phases using a modified, top-down approach. Each phase, along with the results obtained, is discussed below.

Phase One: Unit Testing. Unit testing is the verification of a program module, on an individual basis, to insure that each module functions correctly as an individual unit (18:173). For this effort, unit testing was accomplished using the test plan created during the detailed design phase. A test program was developed which passed parameters to and called module stubs for each module tested. The test program queried the user for the input to serve as passed parameters so the full spectrum of input variables could be tested. The results of each test printed was displayed on the CRT screen for verification. The

results of this test are noted on the design test plan included in

Appendix B, Test Documentation.

Phase Two: Integration Testing. Integration testing is the verification of the interfaces among the systems parts (18:173), focusing on testing the complete program to insure that the interfaces between the modules are working correctly and that none of the modules have an adverse side effect on other modules (3:V-3). Integration testing of the performance monitor was conducted in two steps. Step one involved testing of the on-line monitor. The module "avail packet" was changed so that the recv_packet call would return a hard-coded packet with known values, instead of one from the network queue. This dummy packet was used to test the function of all the modules of the on-line monitor. The results of this test were printed to the screen and verified against anticipated results. All errors were investigated and corrections made. This process continued until the results anticipated matched the results obtained. Then the recv_packet function was changed to return a packet from a test file containing twenty packets. Each packet was read in and the monitor program called appropriate modules to conduct the analysis and display the results, polling for the next packet until all twenty were read and analyzed. The results of this analysis were compared to their expected values and error correction was again conducted. Step Two involved the integration testing of the off-line monitor program. The same file used to generate packet traffic for the on-line monitor served as the message file for the off-line monitor. Again, the monitor was run against this file, the results compared and errors corrected.

Both steps accomplished testing using a modified top-down approach. Groups of modules that accomplished a single function (such as determining the network traffic distribution -- net_site, mat_print, mat_tot, and mat_scrn) were integrated together and tested. When the results achieved were the same as the results anticipated, secondary modules (such as histo_print) were integrated in and tested. This approach allowed the top level module to be first integrated and tested and then the branches of called modules integrated and tested until the operation of all modules was fully verified.

Phase Three: System Testing. System or validation testing is designed to ensure the set of software program performs in accordance with the requirements. In other words, system testing is the process of trying to find discrepancies between the system and its original objectives (18:231). In this case, system testing was accomplished using an artificial traffic generator, a test program used to place traffic with known performance metrics on the network. The results produced by both the off-line and on-line monitors were compared to those calculated for the traffic generator and discrepancies in the results were corrected. The use of an artificial traffic generator program allowed for several categories of tests to be run. These categories are as follows:

A. Load/Stress Testing: This type of testing is used to find the limits of the program being tested. In this study, high traffic volume periods were alternated with low volume periods and the recoverability of the monitor was checked. Also included was volume testing which tested a continuous stream of high volume traffic. The on-line monitor was able to recover from short bursts of high

75

volume traffic very well, storing packets for future processing and signalling when storage space was inadequate for requirements. Additionally, the monitor was able to handle a continuous packet arrival rate of approximately one packet every 2 - 3 seconds. Arrival rates much faster than that soon burdened the storage capacities of the monitor; the measuring process proved unable to keep up with the packet traffic and packets were lost.

B. Storage testing: Conducted in conjunction with load testing, this is designed to test the amount of storage available to the on-line program. Due to limitations in the way storage was dynamically allocated, the exact amount of storage available to the on-line monitor was impossible to determine. Roughly speaking however, after initial program storage allocation (for tables and heads of queues), the system had approximately five thousand characters of storage available.

C. Recovery Testing: This testing tested the program's ability to recover or gracefully degrade in the presence of system or transmission errors. In the on-line monitor errors occurring during initialization caused the program to be terminated, other errors caused error messages to be printed and recovery procedures instigated.

Phase Four: Operational Testing. The final phase in testing is to run the developed software in an operational environment to determine its ability to meet desired objectives. Unfortunately, a fully operational DDBMS was not available for operational testing. Instead, an artificial traffic generator was used to simulate network and DDBMS traffic. This

traffic generator was an extension of the one used in Phase Three described above. In this instance, a greater attempt was made to create a more meaningful artificial traffic generator. Rather than just a program that read a traffic file onto the network, programs were activated at several of the network workstations with the traffic routed between them. It was during this testing that the following problems were discovered.

A. ISO interrupts - The ISO layer 2 software currently implemented on the LSINET is interrupt driven software. Basically, when a packet arrives, the work currently being done is suspended, the interrupt serviced (packet stored to the queue), and work continued. A problem in interfacing the on-line monitor to this software was discovered when running the monitor at full speed. Since the routines that print information to the screen were interrupt based, there was no way to disable interrupts for critical regions of monitor code. Unfortunatel⸱ if a packet happened to arrive while in one of these critical regions, information was lost or the screen garbled. A temporary solution to this problem was to run the monitor on polled, not interrupt driven software.

B. On-line Monitor File Size - The on-line monitor quickly (approximately 30 minutes) filled the message save file with packets when the artificial generator was running full speed (1 packet every 2 - 3 seconds) on several nodes. A solution to the problem of a limited save file space will be to implement the monitor on a system with a hard disk and take off the 1600 packet constraint currently placed on the size of the save file. This is not an urgently required fix since it required all of the traffic

generators to be running full speed to discover the problem and normal, network traffic would not be that heavy.

C. Off-line Monitor File Size - The off-line monitor worked perfectly on small sized message save files, however large files caused insufficient memory conditions in calculating two of the performance metrics, directory access and DDBMS response times. The *reason for this was that the amount of memory required for the calculation was directly proportional to the size of the save file. The software was originally designed this way to allow the metric to be calculated with only one pass through the save file, minimizing the time spent calculating the metric.* Both of these modules were redesigned to require only a fixed amount of memory, regardless of the size of the message save file. This fix, although minimizing the amount of storage required, required several passes through the message save file. The maximum number of passes required equals the number of processes. This is an adequate fix; the only disadvantage noted is the amount of time required to analyze the file for these metrics.

Summary of Chapter Four

In this chapter the installation of the monitor on the LSINET was discussed. The monitor environment was explained and the implementation difficulties explained. The actual performance metrics calculated were included as was a description of the modules developed. The chapter concluded with a discussion of the testing process the monitor underwent. Each phase of testing was presented and the results from the tests conducted during that phase discussed.

V. Recommendations and Conclusions

Introduction

Previous chapters detailed the process of designing and implementing a performance monitor for a distributed data base management system. Specific problems encountered during the study and the solutions adopted were discussed in detail. This chapter is designed to look at the whole process of performance monitoring.   The chapter will first present the overall results of the study, discussing both the on-line and off-line versions of the monitor. Next some conclusions reached on the subject of performance monitoring in general will be discussed and recommendations for future studies presented. Finally, the chapter concludes with a few comments about the project.

Results of the Study

This study accomplished its primary goal of installing an operational DDBMS performance monitor on the DEL LSINET. The monitor as installed, however, is in a preliminary stage and requires much more work to insure that the metrics currently monitored are meeting the requirements of those using the monitor. As it is currently implemented, the monitor programs provided for a very "rudimentary" evaluation of the network traffic. The on-line portion of the monitor is restricted to simple counts (and the statistics derived from these counts) due to the limitations of the hardware. With the network enhancements planned, this limitation will be reduced and the monitor can be made more meaningful. The off-line portion of the monitor concentrates on the evaluation of thruput metrics, less on the areas of response time and utilization. This fact should be corrected.

Conclusions About the Study

During the research phase, a wealth of information was presented on the need for and value of performance monitors. But little documentation was found which provided information on how this evaluation was conducted, what metrics were valuable, what differences existed between distributed and centralized performance evaluation. Only three studies, two of which were conducted by AFIT students, provided useful specifics regarding performance monitoring. As the process of designing and implementing the actual monitor commenced, the reason for this appeared to be that the metrics evaluated are dependent upon the system environment and objectives and are not of a general enough nature to discuss. Some of the articles in more recent publications seemed to bear this out, since they were largely slanted to a specific environment. Although the actual implementation of a performance monitor may be highly dependent upon the environment, general concepts in the area of performance monitoring do not share this dependency. The general process of designing and implementing a performance monitor needs to be formulated, (although the study by Captain Paul Bailor presented an excellent guideline), and sample metrics for different types of networks discussed and evaluated. This leads to the conclusion that not enough work has been done in the area of performance evaluation of a distributed data network.

Future Recommendations

The monitor is in its first stage of development and could be enhanced in several ways. A few suggestions are:

Enhance the Software. The current monitor should be expanded with new metrics calculated as objectives change. Possible new metrics include tracking message formation so that at any time the the percentage of each message received can be calculated and displayed, expanding the utilization and response time metrics for the off-line monitor, deriving a method of calculating network response time, and deriving utilization statistics for each of the network nodes. This could be done by expanding the current centralized monitor to a hybrid and having software packages calculating statistics regarding node performance present on each of the nodes and operating concurrently with the applications software. These nodes could then update the central monitor with the results of the individual monitoring and these results could be incorporated. Finally, the results of the analysis should be upgraded, allowing for graphics displays and hard copy output.

Enhance the Hardware. The software should be transferred to the Intel 310 whenever possible. This system is ideal for the monitor since it allows multiprocessing and could serve as both a node and monitor simultaneously. Additionally it is equipped with a faster processor and hard disk storage allowing for the monitor to run faster and eliminating the time and space constraints discussed in Chapter Four. Additionally, the monitor should be transferred to the Ethernet environment when it is available, eliminating the need for artifacts on the network.

Monitor the Monitor. As mentioned earlier, both the on-line and off-line portions of the monitor have "hooks" installed which will allow for future monitoring of the monitor software. Although the code was designed as efficiently as possible, no attempt was made to monitor,

81

modify, or optimize the code. Additionally, since the code was not
tested against actual DDBMS traffic, this operational testing remains to
be conducted when the traffic is available.

Final Comments

This study has been a fruitful one and the initial objectives were
achieved. Future study should concentrate on a full implementation of
the enhanced monitor on the Intel system.

BIBLIOGRAPHY

1. Air Force Institute of Technology (AU). Development Documentation Guidelines and Standards. Draft AFIT/ENG Standard, revision 2. September 1984).

2. Amer, Paul D. "A Measurement Center for the NBS Local Area Computer Network," IEEE Transactions on Computers, Volume C-31, Number 8: 723-729 (August 1982).

3. Bailor, Capt Paul D. Development of a Data Base Management System Performance Monitor. MS Thesis, GCS/ENG/83D-2. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 83.

4. Benwell, Nicholas. Benchmarking, Computer Evaluation and Measurement. Washington D. C.: Hemisphere Publishing Corporation, 1975

5. Boeckman, Capt John G. Design and Implementation of the Digital Engineering Laboratory Distributed Data Base Management System. MS Thesis, GCS/ENG/84D-5. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 84.

6. Bruner, Capt Timothy D. Continued Development of a Data Base Management System Performance Monitor. MS Thesis, GCS/ENG/84D-6, School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 84.

7. Imker, Capt Eric F. Design of a Distributed Data Base Management System for use in the AFIT Digital Engineering Laboratory. MS Thesis, GCS/ENG/82D-21. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 82.

8. Champine, George A. Distributed Computer Systems. New York: North Holland Publishing Company, 1980.

9. Digital Equipment Corporation. Introduction to Local Area Networks, 1982.

10. Enslow, P.H., Rosen, S., Sammet, J. E., Ferrari, D., and DiNoe, J. Encyclopedia of Computer Science and Engineering Second Edition. Edited by Anthony Ralston. New York: Van Nostrand Reinhold Company, 1983.

11. Ferrari, Domenico, Computer Systems Performance Evaluation. Englewood Cliffs: Prentice Hall, 1978.

12. Hartrum, Dr. Thomas C. Professor. Personal Interview, Air Force Institute of Technology, Wright Patterson Air Force Base, Dayton, Ohio.

13. Hartrum, Dr. Thomas C. and Rowe, Janice F. _Applications Level Monitoring of a Distributed Data Base System with Real Time Analysis._ Unpublished paper, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio.

14. Hartrum, Dr Thomas C. Lecture materials distributed in EE 690, Software Systems Programming. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH July 1985.

15. Kroenke, David. Database: _A Professionals Primer._ Chicago: Science Research Associates, INC., 1978.

16. Lucas, Henry C. "Performance Evaluation and Monitoring," _Computing Surveys_, Volume 3, Number 3: 79-91 (Sept 1971).

17. Mockapetris, Paul V. _Communication Environments for Local Networks._ Contract ISI/RR-82-103. Information Sciences Institute, Marina Del Rey, CA, Dec 1982.

18. Myers, Glenford J. _Software Reliability Principals and Practices._ New York: John Wiley and Sons, 1976

19. Nutt, Gary J. _Computer Systems Monitoring Techniques._ Contract NSF# GJ 660. Department of Computer Science, University of Colorado, Boulder, CO, Feb 1973.

20. Shooman, Martin L. _Software Engineering._ New York: McGraw Hill, 1983.

21. Svobodova, Liba. _Computer Performance Measurement and Evaluation Methods: Analysis and Applications._ New York: American Elsevier Publishing Company, 1976.

22. Tanenbaum, Andrew S. _Computer Networks._ New Jersey: Prentice-Hall, Inc., 1981

23. Wedertz, James A. _Design and Implementation of a Centralized Directory for a Distributed Data Base Management System._ Ms Thesis, GCS/ENG/85D-24. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 85.

24. Woffinden, Capt Steven. Lecture materials distributed in EE 593, Software Engineering. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, April 1985.

VITA

Captain Janice Forsen Rowe was born on 5 February 1955 in Burbank, California. She graduated from La Reina High School in Thousand Oaks, California in 1973 and attended Loyola Marymount University from which she received the degree of Bachlor of Science in Mathematics in June 1977. Upon graduation, she received a commission in the USAF through the ROTC program and was called to active duty in July 1977. She served at the San Antonio Data Services Center, San Antonio, Texas as a Telecommunications Analyst from July 1977 to May 1981. Her next assignment was to the 7102nd Computer Services Squadron, Ramstein AB, Germany and served as a Plans and Programs Officer until entering the School of Engineering, Air Force Institute of Technology, in May 1984.

Permanent address: 1366 East 10th North

Logan, Utah 84321

APPENDIX A

System Environment

Introduction

This appendix presents a detailed description of the functioning of the LSINET, focusing on the hardware employed, communications protocol, and one of the applications programs which use this network, the LSINET DDBMS.

Description of the LSINET

The AFIT Digital Engineering Laboratory (DEL) LSINET is a local computer network located in the School of Engineering building. The LSINET name comes from the use of the LSI-11 microcomputers for many of the nodes. The LSINET consists of 13 micro- or mini-computers interconnected in a star topology as shown in Figure A-1. The LSINET uses the NETOS communications protocol for all message traffic. This protocol was designed to conform as closly as possible to the seven-layer Open Systems Interconnection reference Model developed by the International Standards Organization (22:15-17).

Each micro- or mini- computer in the network is called a node with node B being the central node or system. All of the nodes are linked by full duplex serial communication lines using a subset of the RS-232 standard. The serial links operate at a maximum of 9600 baud. All nodes may communicate with each other by exchanging messages which are routed by and through the central node.

The communication protocal described below is shown in Figure A-2. The basic unit of information transfered can be a file, string, or buffer and is created at the layer 6-7 level. This information field along with the source and destination of the message are passed down to layer 5. This layer is responsible for dividing the information field into blocks,

A - 1
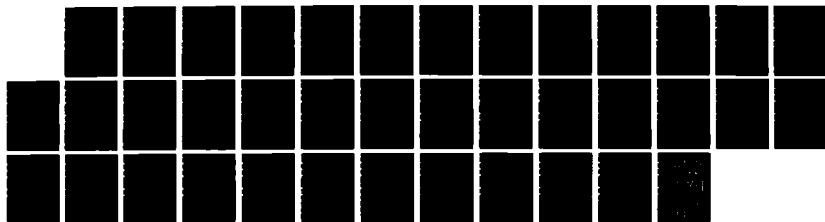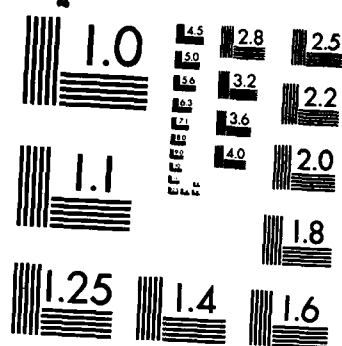
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
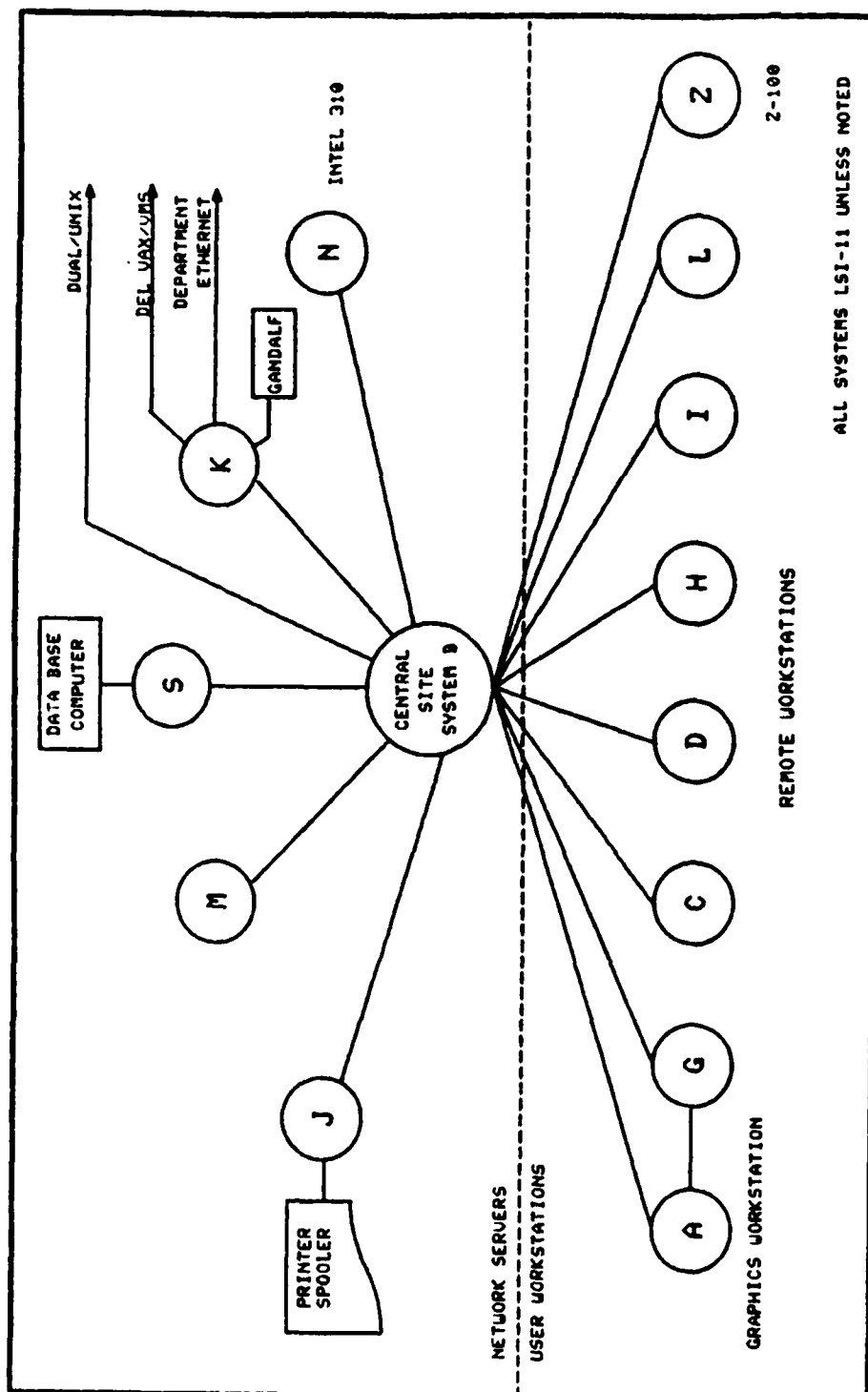
FIGURE A1. LSINET CONFIGURATION

FIGURE A2. NETOS MESSAGE ENCAPSULATION PROCESS

A - 3
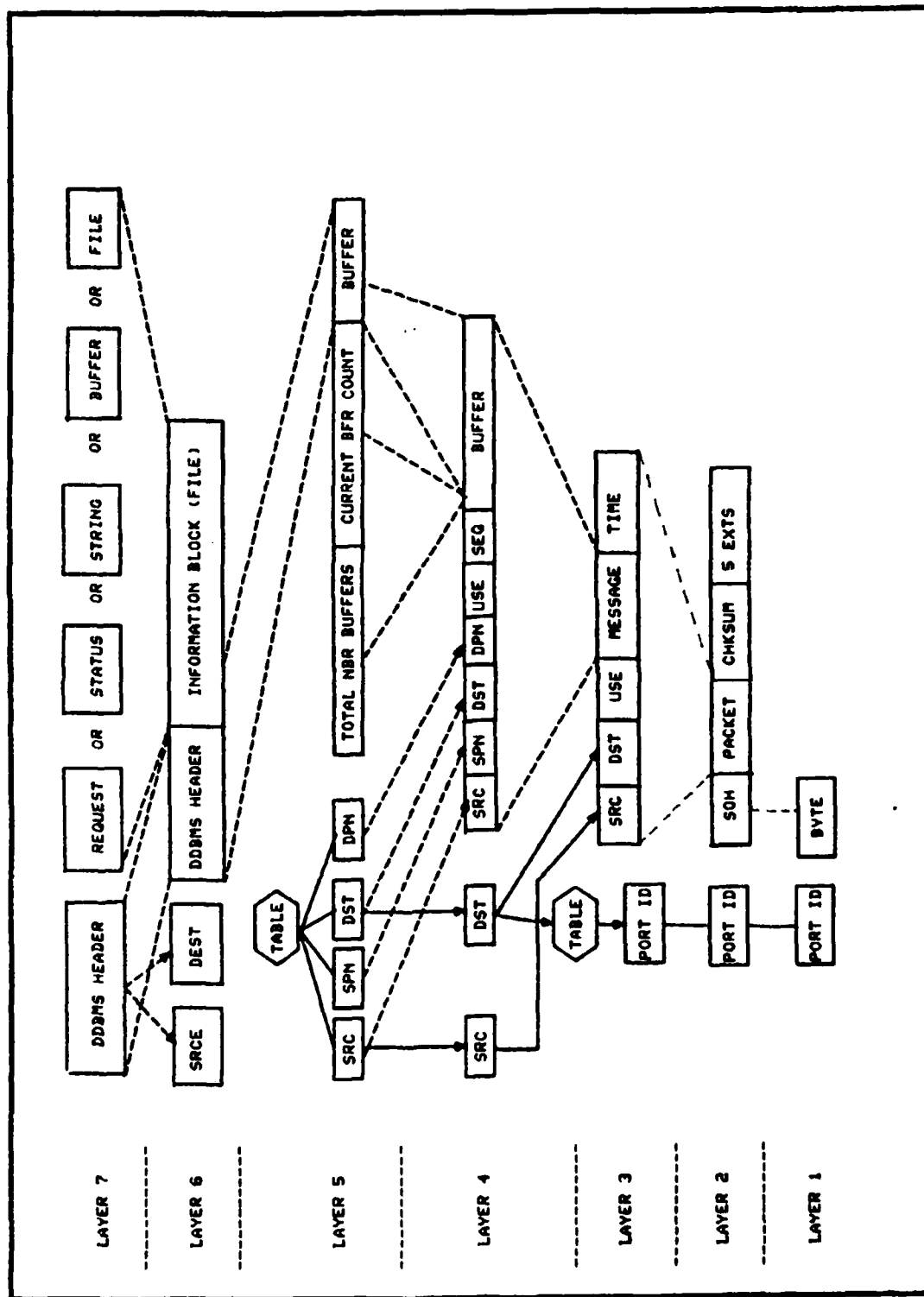
determining how many blocks are to be transmitted and which block is currently being tranmitted. This information is included in a layer 5 header that is passed as part of the data buffer to layer 4. Layer 5 is also responsible for transforming the source and destination codes passed down from layer 6-7 into the network source, destination, source process number and destination process number and passing this information along with the data buffer to layer 4.

Layer 4 is responsible for dividing the data buffer into messages, creating a layer 4 header by adding use and sequence fields to the information passed from layer 5 and passing the entire message buffer to layer 3 for transmittal. Layer 3 takes this message buffer, adds a layer 3 header which consists of a duplication of the source and destination fields, a use field and a timestamp and passes the entire packet to layer 2. Layer 2 calculates a checksum for the packet and builds a frame that includes a start of header character, the packet, the checksum, and 5 end of text characters. This frame is then transmitted, character by character, by layer 1.

When a frame is ready for transmittal, the sending site sends a transmit request to the central site. The central site utilizes polling to determine which of the other nodes requires service. When the central site polls the sending node and sees the transmit request, it sends a transmit acknowledge to the sending site. The frame is then sent and if the central site receives it correctly an acknowledgement is returned to the transmitting node. The central node then determines the destination of the frame sends that node a transmit request, waits for the receipt of a transmit acknowledge and then sends the frame on. If a transmit

acknowledge is not received, the central site will eventually time out delete the packet and continue the polling process. If the frame is transmitted to the destination, an option set in the central site software will allow a copy of the frame to be sent to the monitor program. A more detailed description of the operation of the LSINET may be found in the LSINET documentation (14).

Description of the LSINET DDBMS Messages

This section shows the format for the DDBMS messages transferred over the network. These message formats are derived from the formats of the subset of messages currently under development in a different thesis effort. These formats are a variation of those proposed by Capt Boeckman in the original DDBMS thesis effort. When the monitor program was written, the divider between fields was a CRLF, two characters. Later changes made this divider simply the line feed. Since the message analysis was very position dependent, and it was too late to change the code, the decision was made to keep the divider at two characters, and having the module that reads in the packet modified to add the second character between fields.

DDBMS Header. All DDBMS message traffic will contain the standard
header shown below:

CharValueCount

Char	Value	Count
0	Start of Text - STX	1
1-3	Message Type	3
4	Line Feed	2
5-14	Destination ID	10
15	Line Feed	2
16-25	Source ID	10
26	Line Feed	2
27-30	Unique Process ID	4
31	Line Feed	2
32-41	Time Stamp	10
42	Line Feed	2
42-N	Message Dependent	N-42
N+1	End of Text	1

DDBMS Messages. All of the message types detailed in Boeckmans effort
are considered valid even though they are not currently implemented. Not
all of the messages are monitored for however. The following is a list of
the DDBMS message types currently being monitored for by the network and
their meaning. Other message types, although equally valid, are not
monitored specifically. Instead they are treated simply as any other DDBMS
message.

Message	Message Type
DDBMS Ready Command	DRC
Deleted Site Message	DSM
Added Site Message	ASM
External deleted site command	EDS
External added site command	EAS
Unique process started	BEG
Unique process ended	STP

Appendix B

Test Documentation

Contents

TABLE B-I

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Control Monitor (0.0)	Central system activated and passing messages	Run Program	Query user for next action	Not Implemented
		Action = run online	Call online analysis module	Run online monitor
		Action = run offline	Call offline analysis module	Run offline monitor
		Action = exit	Exit program	Not implemented
Online Analysis (A0.0)	Central system activated and passing messages	Call module	Call initialization modules	Variables & struct initialized
			Query user for output format	Entry to query screen results in correct format
			Print message and Poll for next pkt	As expected
		Status flag = PKTNAVL	Print message and Poll for next pkt	As expected
		Status flag = PKTAVL	Call xfer_pkt module	As expected
		Status flag = HDRAVL	Call anlyz_pkt module	As expected
		Status flag = HDRNAVL	Poll for next pkt	Polling correctly resumes
		Status flag = MEMERR	Call fix_memerr module	Allocated space properly freed
		Status flag = IRRERR	Print message and terminate program	As expected

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Online Analysis (A.0.0) (Continued)		Done flag = 0	Continue operation	As expected
		Done flag = 1	Return to calling module	Exit Program
		Done flag = 2	Open file DDMON.MSG	Open DDMON.MSG Reset saved pkt count and continue operation
		Saved pkt count = 1600	Close DDMON.MSG and print message	As expected
Init_qryscn (A.1.0)	Online analysis module running	Call Module	Print menu to CRT and get user action	Menu printed and action verified
		Action = A	Call scrn_one_prnt Set scrn_one_flag = 0 Return done = 0	As expected
		Action = B	Call scrn_two_prnt Set scrn_flag = 1 Return done = 0	As expected
		Action = C	Return done = 2	As expected
		Action = X	Return done = 1	As expected
Scrn_one_prnt (A.1.1)	None	Call module	Format screen for primary results output	As expected

TABLE B-I (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Scrn_two_prnt (A.1.2)	None	Call Module	Format screen for active/inactive site parameter output	As expected
Init_online (A.2.0)	Online Analysis Module Active	Call Module	Call init_var to initialize variables	Variables properly initialized
			Call init3 to start ISO level 2 packet receive function returning status	As expected status values correct
		Status = true	Return status = IRRERR	Correct value returned
		Status = false	Call ddtabl_build module which returns status	Module called and ddtabl correctly built
		Status = IRRERR	Return status	As expected
		Status = false	Call msgtab_build module which returns status	Module called, msgtab correctly built, status values correct
		Status = IRRERR	Return status	As expected
		Status = false	Call queue_init module which initializes struct and returns status	Structures correctly initialized and status values correct
			Return status to calling module	Status returned to Calling module

TABLE B-1 ((Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Init_var (A.2.1)	Global variables need to be initialized	Call module	Variables set to initial values	Values verified and algorithms corrected
Init3	Program linked with modified ISO level 2 & 3	Call module	Initialized ISO level 2 & 3 variables and start level 2 pkt queue and return error conditions	As expected
DDTabl_build (A.2.2)	DDTAB.DAT file exists on disk	Call module	load DDTABL from floppy disk as linked list return proper error codes	As expected Note: No error return when file DDTABL.DAT corrupted
Msgtab_build (A.2.2)	MSGTBL.DAT file exists on disk	Call module	load MSGTAB from floppy disk as a linked list and return errors	As expected (Same note as above)

B - 5

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Queue_init (A.2.4)	None	Call module	Initialize pointers to the head of all queues (linked list) return error codes	Pointers properly initialized Next rec pointers set to NULL
Fix_memerr (A.3.0)	Out of Memory error (MEMERR) occurred	Call module	Re-initialize queues free allocated storage and return	As expected
Avail_pkt	Packet storage area allocated	Call module	Call ISO level 2 recv_pkt which returns pkt or error Return PKTNAVL	As expected
		Error = NO MSG	Return PKTNAVL	As expected
		Error = false	Decrement lvl_2q count and return PKTAVL	lvl_2q count decremented and qtimer read and used to update current time and time last pkt rcvd Display screen updated and return status = PKTAVL

B - 6

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Xfer_pkt (A.5.0)	Packet available	Call Module	Call det_pkt_type module which returns status	As expected
		Status = NETPKT	Not anticipated	return status and resume polling
		Status = DDPKT	set up current pkt and total pkt count then call xfer_queue which returns status	As expected
		Status = HDRAVL	Call parse_to_permq	As expected
				Return status to calling program
Det_pkt_type (A.5.1)	Packet available DDTABL created	Call module	Set up comparison SRC & SPN and call search_ddtab which returns location status	As expected. Return status to calling module
Search_ddtab (A.5.1.1)	Packet available DDTABL created Values for host and process number provided	Call module	Search DDTAB for a matching host and process number Return DDPKT if match or NETPKT if no match found	Check for use = 1 If yes return NETPKT the rest functioned as expected

TABLE B-I (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Xfer_queue (A.5.2)	Packet available with type equal DDPKT	Call module	Transfer data from packet to tempq, check if DDBMS hdr is complete and return status	As expected
Adjust_pkt table (A.5.3)	Packet available	Call module	Search pkttab for packet originating message and adjust counts accordingly When counts equal reset message	As expected
Parse_to_ permq (A.5.4)	Complete tempq entry available	Call module	Parse out the entry in the tempq to permq fields, free tempq storage and return	As expected except to insure tempq next record pointers reset before freeing entry
Anlyz_pkt (A.6.0)	Entry available in permq	Call module	Search msgtab and analyze permq entry according to status returned, update results and return	As expected

B - 8

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Update_rslts (A.7.0)	Scrn_flag set	Call module	Based on value of scrn_flag, update display with value in results record	As expected
Offline Analysis (B.0.0)	DDMON.MSG file exists on disk	Call module	Print user actions menu to screen and get next action	As expected
		Action = A	Call det_thru_put module	As expected
		Action = B	Call det_resp_time module	As expected
		Action = C	Call det_dict_acss module	As expected
		Action = X	Return to the calling program	Exit program
det_thru_put	None	Call module	Print user actions menu to screen and get next action. Set mode and done flags appropriate to the action and call the proper module	As expected

B – 9

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Net_Site (B.1.1)	Mode set in calling module and DDMON.MSG file on disk	Call module with mode = NETWORK	Analyze all packets in DDMON.MSG file	As expected
		Call module with mode = DDBMS	Analyze only DDBMS type packets from DDMON.MSG file	As expected
		Complete pkt available	Determine source and destination and increment count fields accordingly	As expected
		End of File	Query user for Display formats and call appropriate modules	As expected. End of module shown by X entry in query
Mat_print (B.1.1.1)	Completed net_site analysis with updated matrix record	Call module	Query user for action	As expected
		Action = A,B, C or D	Set tsrce &tdest per action and call mat_scrn_paint module query usr for action	As expected
		Action = E	Call tot_scrn_paint and query user for action	As expected
		Action = X	Return to calling module	As expected

TABLE B-1 (Continued)

Module level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Mat_scrn_paint (B.1.1.1.1)	Tsrce, tdest and mode are passed, matrix records have been updated	Call to module	Print the selected quadrant of the traff dist matrix from the values in matrix records and return	As expected but give the option to reprint screen
Tot_scrn_paint (B.1.1.1.2)	Mode, total count and matrix records are available	Call to module	Print the info in the totals matrix matrix and return	As expected
Histo_print (B.1.1.2)	Updated matrix records and mode available	Call to module	Starting with SRC & DST = A, Determine if a link had any traffic, if so print histogram to screen. Return when SRC & DST = Z	As expected but give user option to quit after each screen display or review the display before exiting
DD_msg (B.1.3)	DDMON.MSG file available	Call to module	For all packets of type = DDBMS, det msg type, add to count and display results when EOF	As expected

B - 11

TABLE B-I (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
net_time	DDMON.MSG file is available	Call to module mode = NETWRK	Analyze all traffic	As expected
		Call to module mode = DDBMS	Analyze only DDBMS Traffic	As expected
		Complete pkt available	Determine hour pkt was received and increment correct count	As expected
		End of File	Print totals to screen by time until user enters X then return	As expected but divide output by page not scroll
Det_dict_acss (B.3.0)	DDMON.MSG file available	Call to module	Check for DDBMS pkts and partially reconstruct DDBMS msgs to determine directory access. Calculate access for each directory by process and update results records Upon completion of all processes print results to screen and exit	As expected after correcting algorithms

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
DD_xfer_acss (B.3.1)	Computations for a single process complete	Temporary calculations record	Based upon the value in the flags, update the results records with the number of dictionary accesses for local, central and extended dirs	As expected
Acss_rslts (B.3.2)	Updated results available in the rslts record	Call to module	Print the results from the rslts record to the screen	As expected, minor reformatting required
DD_resp_time (B.2.2)	DDMON.MSG file available	Call to module	For all packets of type DDBMS, search for complementary message types (BEG-END, RQR-RQM, CDR-CDM etc) and compute time difference and store by process When complete for all process numbers, Calculate average response times and display results	As expected but needed to assume synchd clocks and change design so more than one msg of the same type per process number could be computed

B – 13

TABLE B-1 (Continued)

Module Level Test Plan

MODULE	CONDITIONS	INPUTS	EXPECTED RESULTS	ACTUAL RESULTS
Init_wrk_rec (B.2.2.1)	File DEMON.MSG open	Call to module	The next unique process number from the file will be stored in the wrkrec and status = process found. All process numbers completed status = done Return status	As expected
Prnt_rslts (B.2.2.2)	Updated rslts available	Call to module	Print the results of the analysis to the screen and return	

B - 14

Integration Testing

Step One: Integrate Analysis Modules

In this step the both the on-line and off-line analysis modules were merged to form the on-line analysis program and the off-line analysis program. Problems of the following types were encountered.

A. Redundant variable/module names -- minor difficulties were discovered in that some of the module and variable names were identical in the first five characters and the error message "re-declared variable VARNAME" was received during compile. Changing the names solved the problems.

B. Module calls and returns -- some problems were encountered when calls to modules had the passed variables in the incorrect order or failed to have the correct number of passed variables. All modules were checked and the problem resolved.

C. Declaration of Modules -- one of the largest problems encountered was the fact that the default value of module returns is integer if the module is not declared regardless of the fact that the return value is declared properly. This happened with the call to the function in the Qlib where the returned value was a long decimal as was the variable that received the returned value but the function was undeclared. After declaring the function, the problems created were resolved and all the modules that returned values were likewise declared.

Step Two: Test with hard-coded packet

The next step involved testing the on-line analysis program against a single hard-coded packet. A 142 character packet was created in the module avail_pkt and returned to the program to be analyzed. The main problem discovered was incorrect field position when parsing the tempq into the permq.

Step Three: Testing with an on disk packet file.

This step changed the avail_pkt function to read a packet from a test file containing twenty packets all of type DDBMS. No major difficulties were encountered but minor problems in algorithm design were discovered and corrected. Both the on-line and off-line programs were tested against the same packet file.

Step Four: Test with network traffic

The final step in integration testing involved testing the on-line analysis program against actual network traffic. Network traffic was generated by the existing DEMO3 programs and no major problems were discovered. In this testing, the packet stream was not continuous. The packets were sent one at a time and the operation of the on-line monitor verified through the use of readkeys.

System Testing

Step One: On-line Analysis Program

In this phase, the on-line analysis program was operational and DDBMS traffic simulated via an artificial traffic generator which sent a constant stream of DDBMS traffic across the network. All of the readkey functions that allowed the user to slowly step through the on-line analysis modules were turned off (using preprocessor statements set to zero) and the same packet file used in integration testing was used. The major problem discovered involved the interrupt handling of a received packet while the program was involved in a critical area of code. The receipt of a packet at this time would cause non-predictable, non-standard results. Either the program would Ftrap out (if in the middle of a calculation) or garble the print screen (if in a printf statement). An attempt was made to solve this problem by disabling the interrupts when a critical area of code was entered and enabling them when the area was exited, but unfortunately too many of the standard C functions like printf use the interrupts and disabling them caused the program to abort. This problem was resolved by using the polled version of the ISO2 software but it remains to be solved.

Step Two: Off-line Analysis Program

In this phase, the off-line analysis program was run against the DDMON.MSG file created by step one. A major difficulty was discovered in the reading and writing to files. As originally designed, the on-line

B - 17

program was to have written to the files in a character-by-character fashion, using the function putc. Similarly, the off-line program would have read the file using getc. The putc function however does not work with NULL values and no error message is printed (Saying unable to write etc.) but the problem was discovered by calculating the packet length. The solution to the problem was relatively simple. The received packet was logically or'd with the value 0100H on a character by character basis before storing to the file with an fprintf command. This changed the character to a 16 bit value greater than 0. Upon reading the character, it was again logically and'd with the value 0011H which transformed the character to its original value.

Operational Testing

Due to the non-availability of actual DDBMS traffic, true Operational Testing could not be conducted. However, the artificial traffic generation program was expanded to provide a more realistic traffic load (with pauses and by sending out packets with sequence numbers 0 thru 3). While the traffic generator created the DDBMS traffic stream, the DEMO3 program was loading the network with simple network traffic from other nodes. Generally, the on-line monitor could handle these adequately, taking approximately 1 second to process a DDBMS packet, .6 second for a network packet. The main problem was that the traffic, with generators running at full speed and transmitting a packet every two to three seconds, soon filled the DDMON.MSG file causing the out of file space message to occur. The on-line monitor continued to operate, although packets were not saved for off-line analysis.

A major problem with the off-line monitor was the out of memory condition that occurred when the DDMON.MSG file size was maximized. Although the logic was sound, some of the modules were redesigned to minimize the amount of storage they required in order to process the very large message files. No other major problems were encountered.

APPPENDIX C

Documentation Tools and Techniques

Contents

Page

Introduction

This appendix describes the format for the documentation tools and techniques used in this study. These tools and techniques were used in several classes taught at the Air Force Institute of Technicology and are proposed Engineering Department Standards (1). The specific descriptions of thes tools were obtained from this standard as well as the study by Capt Paul D. Bailor and are included here for completeness.

Data Flow Diagrams (3: II-26-27)

The mechanics of a Data Flow Diagram (DFD) are shown in Figure C-1. An input data source provides data to a data transformation process. The transformation process converts the input data into output data which is provided to the data sink. A transform process can have more than one input data flow and produce more than one output data flow. Additionally, the transform process can access data files or data bases while performing the data transformation process.

Data Flow Diagrams are intended to show the steady state flow of data within a system with no consideration to control paths such as loops; hence, loops appear very seldomly in DFDs. One situation where loops may occur is an iterative testing process such as hypothesis testing. In these situations, data and test conditions can be modified several times to give a broader range of results. Showing data flow through a hypothesis testing process would naturally seem to require some indication of an iterative porcess or loop. Since loops are seldomly contained in DFDs, there is no agreed upon convention for showing a loop. For this study, a data flow constructed of dashed lines is used to show a loop in the data flow paths,
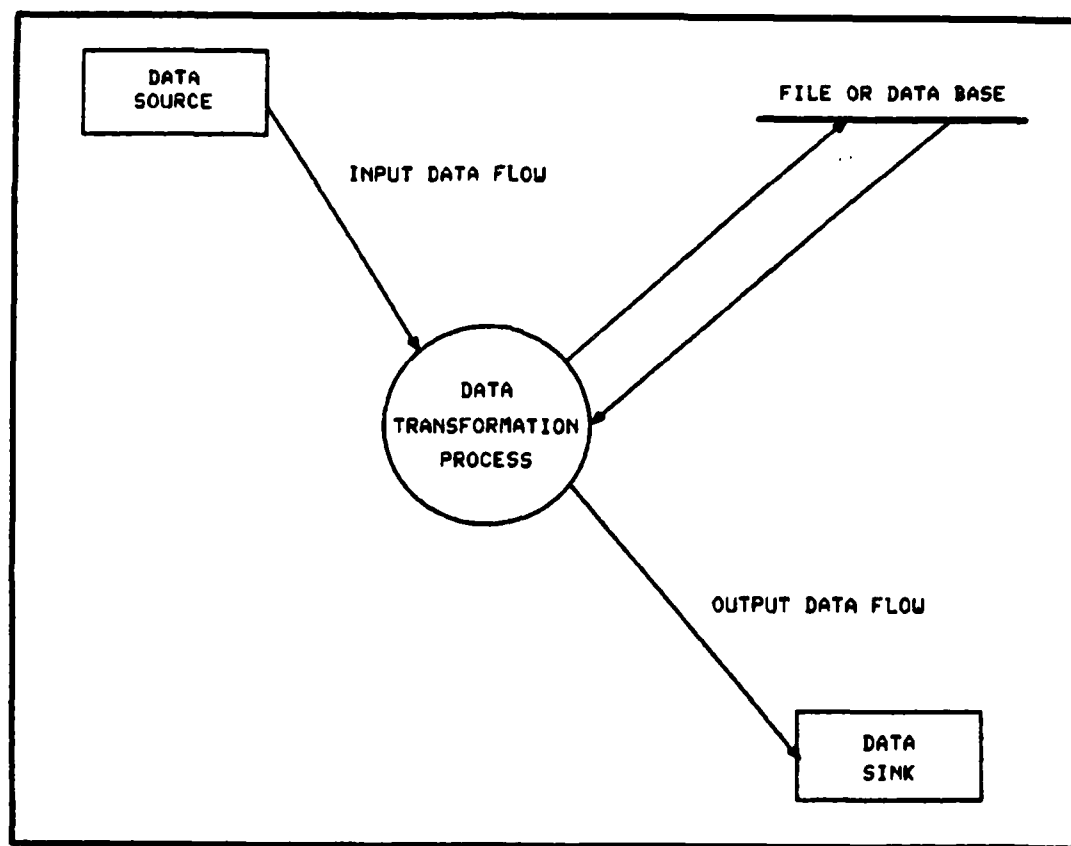
FIGURE C1. DATA FLOW DIAGRAM
(3:II-6)

and in keeping with the intent of DFDs, this convention is used only when absolutely necessary.

Structured Analysis and Design Technique (3:III-1/3)

The Structured Analysis and Design Technique (SADT) developed by SofTech Corporation was used to document the preliminary design. By showing the system activities, the SADT technique specifies what has to be accomplished before the details of how it is accomplished are introduced. Therefore, the implementation details are forced to the lower levels of the problem solution, and the design documentation does not resemble programming logic.

The mechanics of an SADT activity diagram are shown in Figure C-2. The box represents the activity to be performed, and the arrows represent the data associated with the activity. An advantage of SADT activity diagrams is their ability to specify control and mechanism inputs as well as input and outputs. Documentation for SADT diagrams consists of a node index, the activity diagrams with facing page text and a data dictionary.

Structure Charts (1)

A structure chart is used to document the modules used in developing the programs code. The mechanics of a structure chart are shown in Figure C-3. The box represents the process (module). Each should contain the module name and number. A module that has already been identified in another drawing should contain the original module number and be flagged as a common module with a small, filled in triangle in the lower right hand corner. The boxes should be connected with vectors. These vectors are used to connect parent modules with the modules they call (children).

CONTROL

INPUT ——→ ACTIVITY ←—— OUTPUT

MECHANISM

INPUT - DATA CONSUMED/TRANSFORMED BY THE ACTIVITY

CONTROL - DATA WHICH CONSTRAINS THE ACTIVITY

OUTPUT - DATA PRODUCED BY THE ACTIVITY

MECHANISM - PROCESSOR/TOOL USED TO
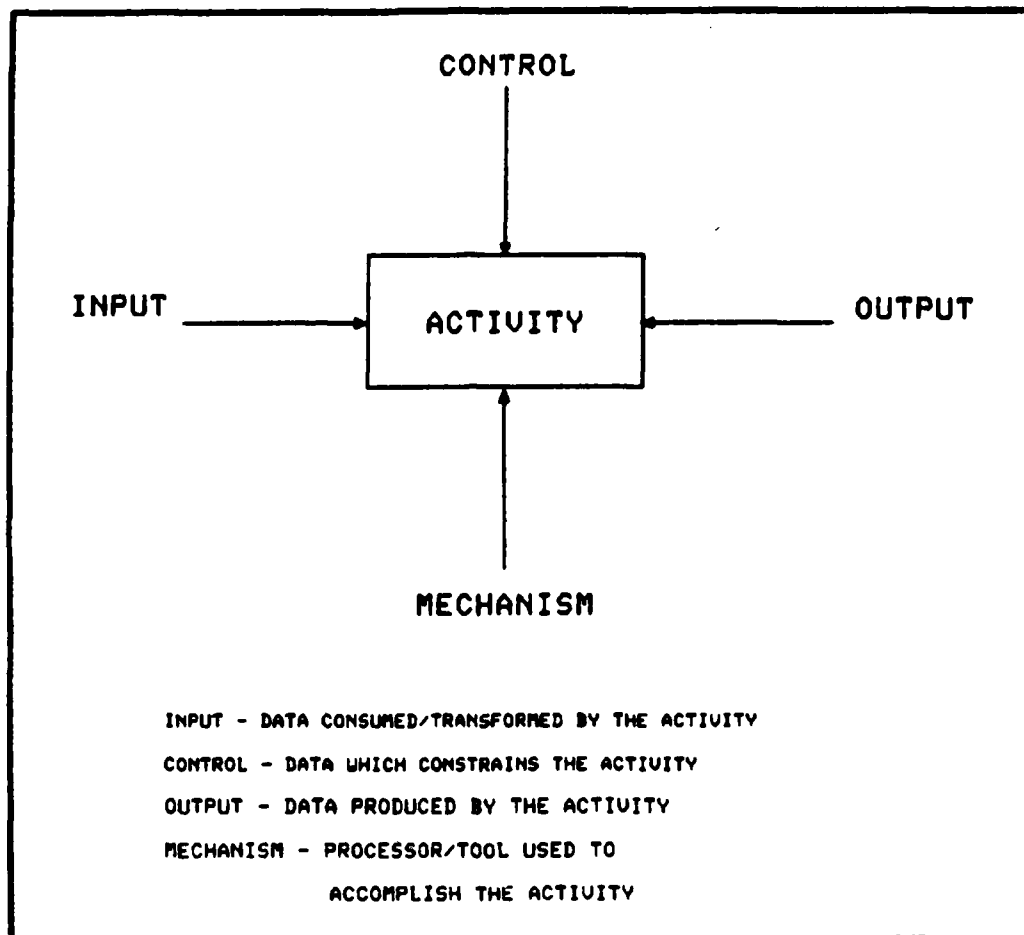   ACCOMPLISH THE ACTIVITY

FIGURE C2. SADT ACTIVITY DIAGRAM
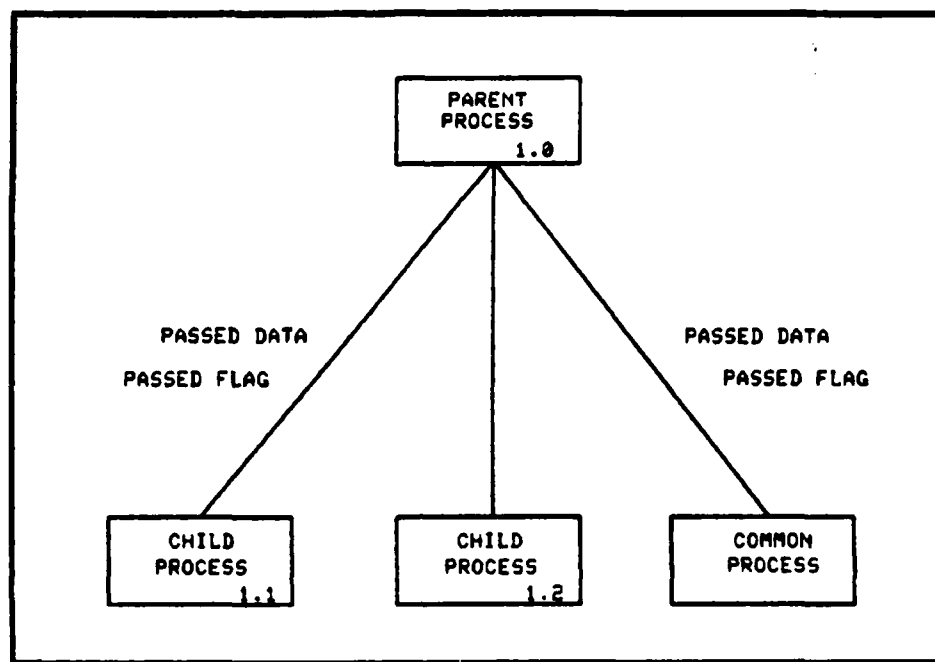   (3:III-3)

C - 5

FIGURE C3. SAMPLE STRUCTURE CHART

Appropriate arrows and parameter names should indicate the parameters passed between parent and child. A single structure chart should be horizontally oriented and show a parent module and one level of children. A second level of children may be shown if it is the bottom level and there are only a few modules.

Monitor Installation on an Intel 310

Introduction

This appendix presents a description of the current status of the implementation of the network monitor on the Intel System 310. The Intel system is a powerful mini-computer, capable of operating in a multi-user, multi-task environment. For this reason, it was chosen to host the network monitor. It would be an ideal performance monitor, capable of running the real-time performance monitor in the foreground, and the detailed performance monitor in the background. Due to a lack of easily understandable documentation and constrained by time, the attempt to use the Intel system was halted and a working LSI-11 system substituted as the monitor hardware. The use of the Intel 310 for the monitor should be pursued however, and this index will provide useful details on the current status of the machine.

Hardware Description

The Intel System came with 256K of memory. An additional 512K memory board was installed bringing the total memory to 768K. This memory board is required to run the Intel system with the RMX 86 operating system. Also installed was an Intel 544A communications board. This board is equipped with an 8080 processor and is currently configured to allow up to four serial devices and one parallel device to be attached to the ports. The internal cabling to support the serial connections (ports J5-J8).

Software Description

The RMX 86 operating system has been installed on the system as have
the languages PLM, Assembly, and C. The Intel system boots from a file
located in the directory :sd:/system/rmx86. This file is created by the
user invoking the ICU program (to invoke the user must be in the
/rmx86/icu directory and type the command ICU86), modifying the contents,
saving and regenerating using the ICU commands, exiting the ICU and then
copying the file created from the boot directory to the /system/rmx86
directory. The boot file is currently configured to activate the 544
driver on one of the 544 ports.

Proposed Direction

Two major obstacles prevented using the Intel as the network
monitor, lack of usable documentation and lack of time. The major
problem was the documentation provided. The RMX 86 operating system is
very powerful, but unfortunately very complex. The documentation is
likewise very complex and somewhat obscure. To work with the system, a
thorough understanding of the operating system is required since none of
the languages seem to allow direct access of the ports (all port access
is made through calls to the operating system). Recommended is
attendance at an Intel school that teaches the operating system and how
it interfaces with the Intel 310 as well as a detailed reading of the
documentation provided.

A recommended plan of attack is to first get the system operational
with a super user and two or three of the 544 ports activated. (This can
be done using the drivers provided in the ICU configure file). After the
ports are activated, programs can be written that try writing things to

and receiving data from each of the ports until familiarity with the
system calls is achieved. Finally, the network monitor and the related
NETOS software needs to be brought on-line and modified as required.
Some problems may be encountered in trying to properly configure the 544
driver (which in turn programs the 544 board). The driver allows three
input modes, transparent, normal and flush which are explained in the
configuration guide, page 10-110. None of these seem suitable for
receiving interrupts from a communication line, reacting to them, and
then returning to the working program. A specialized driver may need to
be written.

As envisioned, the 8080 processor on the 544 board would be
responsible for handling the communication links with the LSINET,
receiving the packets and storing them in the memory shared with the
Intel 310 8086 processor. This processor would be responsible for
reading the packets from the memory locations and analyzing them
appropriately. This suggests a carefully planned interface since both
processors could not access the shared memory at the same time and a
method of marking the packets needs to be developed. An easier solution
would be the implementation of a 534 board which does not have its own
processor or an ethernet card (both boards are either available or on
order).

ADA 164129

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	

4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/85D-14	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Paterson AFB, Ohio 45433		7b. ADDRESS (City, State and ZIP Code)

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS.

	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE (Include Security Classification)
See Box 19.

12. PERSONAL AUTHOR(S)
Janice F. Rowe, B.S., Captain, USAF

13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 December	15. PAGE COUNT 130

16. SUPPLEMENTARY NOTATION

17.	COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Networks, Database, Computer Performance Monitor,
09	02		Distributed Database Management System, Computer Program

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A NETWORK MONITORING FACILITY FOR A DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

Thesis Chairman: Thomas A. Hartrum, Phd.,
                   Professor of Electrical Engineering

Approved for public release: IAW AFR 190-1/.
Lynn Wolaver   16 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas A. Hartrum, Phd.	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3576	22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.

Abstract

This investigation ~~thesis~~ designed and implemented a hybrid network monitoring facility on an existing Distributed Data Base Management System. Analysis of the performance evaluations goals and objectives for the complete distributed system (both the layered protocol network and distributed data base) was accomplished. Two monitoring programs were developed. The on-line analysis monitor is designed to work with existing software to calculate metrics involving arrival rates, packet counts and arrival time. The off-line analysis monitor, using the packets saved during the on-line session, completes a more detailed analysis, providing user selectable metrics in the area of throughput, response times and utilization. Both programs were extensively tested using a four phased process which encompassed unit level, integration, systems and operational testing. Operational testing was accomplished using an artifical traffic generator program, designed to produce realistic network traffic. Keywords?

END

FILMED

3 - 86

DTIC